UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

DESIGN OF VARIABILITY COMPENSATION ARCHITECTURES OF DIGITAL CIRCUITS WITH ADAPTIVE BODY BIAS

by

ANAND SIDDHARUDH SANMUKH

in partial fulfillment of the requirements for the degree of

Master of Science
in Electronics

at the Universitat Politècnica de Catalunya. BarcelonaTech,

Supervisor: **Prof. FRANCESC MOLL (francesc.moll@upc.edu)**

ACKNOWLEDGMENTS

I would like to express my gratitude to my supervisor Prof. Francesc Moll for the useful comments, remarks and engagement through the learning process of this master thesis. I would like to thank my family,friends and loved ones, who have supported me throughout entire process, both by keeping me harmonious and helping me putting pieces together. I will be grateful forever for your love.

ABSTRACT

The most critical concern in circuit is to achieve high level of performance with very tight power constraint. As the high performance circuits moved beyond 45nm technology one of the major issues is the parameter variation i.e. deviation in process, temperature and voltage (PVT) values from nominal specifications. A key process parameter subject to variation is the transistor threshold voltage (V_{th}) which impacts two important parameters: frequency and leakage power. Although the degradation can be compensated by the worst-case scenario based over-design approach, it induces remarkable power and performance overhead which is undesirable in tightly constrained designs. Dynamic voltage scaling (DVS) is a more power efficient approach, however its coarse granularity implies difficulty in handling fine grained variations. These factors have contributed to the growing interest in power aware robust circuit design.

We propose a variability compensation architecture with adaptive body bias, for low power applications using 28nm FDSOI technology. The basic approach is based on a dynamic prediction and prevention of possible circuit timing errors. In our proposal we are using a Canary logic technique that enables the typical-case design. The body bias generation is based on a DLL type method which uses an external reference generator and voltage controlled delay line (VCDL) to generate the forward body bias (FBB) control signals. The adaptive technique is used for dynamic detection and correction of path failures in digital designs due to PVT variations. Instead of tuning the supply voltage, the key idea of the design approach is to tune the body bias voltage by monitoring the error rate during operation. The FBB increases operating speed with an overhead in leakage power.

May 24, 2016

CONTENTS

LIST OF FIGURES

1

INTRODUCTION AND OUTLINE

1.1. INTRODUCTION

In the last decade, the computational and speed of mobile and hand-held devices has witnessed great improvements. Performance demanding applications such as 3-D graphics, audio video, internet access and gaming which were used solely for computer systems are now available for mobile platforms too. This is evident in the evolution of the hand-held devices, as in the last decade there is a significant improvement in the speed . Indeed the surge in the market of smart phones, other electronic applications which uses the mobile internet devices and ultra mobile personal computers is expected to push the performance threshold of the processors in the coming years.

The industry has continued to push technology scaling and performance at the rate dictated by the Moore's law [8]. By shrinking transistor dimensions, designers can deliver consistent improvement in computational capability of processors through higher integration levels and faster switching times [9]. Hence the technology scaling has been the fundamental factor for the growth in semiconductor industry.

The parameter variations like Process,Voltage and Temperature often called as PVT variations [10] becoming a serious factor which is degrading the device performance. Because of this there are differences between chips even though they are identical both in design and in process. Variations makes the designing process very stringent and difficult as they have to work under a range of parameter values.

The thesis proposes a adaptive body bias (BB) system to mitigate the PVT effects in a efficient manner. The main test system used includes a Canary logic unit which can generate a predictive warning signal. The warning signal from the canary logic is monitored during a specified time period. According to the warning signal, circuit speed is tuned through body bias such that the performance degradation is compensated. Similarly If no warning signals are generated during the monitoring period, the circuit is slowed down to reduce power usage. The body bias (BB) block is used to generate *forward body bias(FBB)* signal and *No body bias* control signal which goes to the inputs of charge pump. Depending on the control signal the charge pump generates the corresponding FBB voltage to speed up the circuit or it goes to No BB (NBB) to slow down the circuit. Depending on this voltage generation the body bias of VCDL and main circuit is controlled. A large delay difference

between VCDL output and the reference pulse signal indicates the circuit to speed up and it is in FBB mode. Similarly opposite for the no body bias mode. Finally this phase detector achieves its steady state when the VCDL output rising edge is produced at the same time as the reference signal falling edge, and thus VCDL nominal end-to-end delay (corresponding to the reference pulse width) is achieved. Consequently, the circuit operates at appropriate speed according to process, supply voltage, and temperature (PVT) variations, which enables much more energy-efficient operation than the worst-case with guard banding. The proposed system has the following advantages:

- The design approach can be applied to circuit level instead of the chip level approach [11][12]. In other words, each block can be tuned according to its own degree of variation. Evidently, the finer granularity control allows improved power efficiency.
- We no need to have any critical path replica circuit of the actual operating to detect the performance degradation as in the methods [2]. Our approach of detection is more direct and reliable as it shares the similar PVT variations of the operating circuit.
- Its proactive nature can avoid the complex error correction schemes in retroactive systems [13]. The retroactive systems rely on pipeline flush [3] or instruction replay [14] and therefore are restricted to processor designs. In contrast, our system can be applied to both processors and general sequential circuits.
- Its use of DLL based approach to generate control signals which produce the body bias voltage and avoids the use of DAC [15][2][16].

1.2. OUTLINE OF THE THESIS

The thesis is organized as follows. The second chapter provides an introduction to the basics of PVT variability problem and its sources of variations. Next in this section the traditional and adaptive design approaches are described. In third chapter a brief introduction to 28nm UTTB-FDSOI technology is given along with the advantages and the body bias techniques used. In chapter 4 the state of the art typical design cases are discussed which tells about the types of techniques used for the error correction and detection. In thesis proposal the error detection circuit used is canary logic, hence advantages are discussed in this section. Chapter 5 explains the timing analysis and the most important parameters measured. In timing analysis section the different timing paths considered during simulation is discussed and parameters like combinational delay (t_{pd}) , set-up time delay (t_{setup}) and propagation delay (t_{pcq}) measurement is discussed. These timing parameter helps to calculate the minimum clock frequency required for the operation of the system.

In Chapter 6 and 7 the thesis design approach and the simulation results are explained. Finally the thesis is concluded in chapter 8.

2

VARIABILITY PROBLEM

2.1. INTRODUCTION

In general, there are three different sources of variation two environmental and one manufacturing:

- Process Variation
- Supply Voltage
- Operating Temperature

The variation sources are also known as Process, Voltage, and Temperature (PVT) [10]. The aim of the designers should be such that the circuit should operate over all worst cases of these PVT variations. Failure to do so causes circuit problems, poor yield, and customer dissatisfaction. In the next paragraph let us have some insight on the statistical distribution. Variations are usually modeled with uniform or normal (Gaussian) statistical distributions, as shown in Fig.2.1. Uniform distributions are specified with a half-range. For good results, accept variations over the entire half-range. For example, a uniform distribution for VDD could be specified at 1.0 V ,±10%. This distribution has a 100 mV half-range. All parts should work at any voltage in the range. Normal distributions are specified with a standard deviation σ Processing variations are usually modeled with normal distributions.

(a) Uniform distribution

(b) Normal distribution

Figure 2.1: Uniform and Normal distribution derived from [1]

Fig.2.2 shows the frequency measured and leakage for 62 dies [2]. The variation is due to Die-To-Die(D2D) and Within-Die(WID) process variation. From the Fig.2.2 the vertical dashed line represents the minimum frequency requirement for each die. Assuming a worst-case power density depending on the design and application, the maximum allowed leakage power for each die can be calculated using the die frequency, switched capacitance and a worst-case activity factor(α). Any die at its maximum allowed operating frequency, if the leakage power exceeds the maximum, then the die must be either accepted at a lower operating frequency, or discarded if the standby leakage power I_{SB} limit is exceeded. A significant number of dies become unacceptable since they fail to meet one of these two constraints.

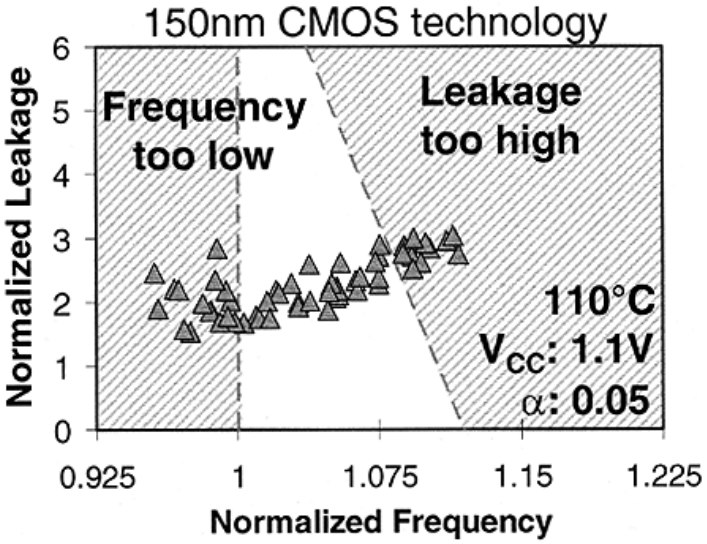

Figure 2.2: Measured leakage power and frequency for 62 dies derived from [2]

2.2. CATEGORIZING SOURCES OF VARIATIONS

The source of variations can be *local* or *global* affects all the transistors on the die. For example, voltage fluctuation by Power Supply Unit (PSU) affects the entire die. Another example for this variation is Inter-die variations. Contrary to global sources of variation, local effects are limited to a few transistors in the immediate vicinity of each other. Examples of local variation include voltage variation due to resistive drops in the power grid, etc. Variations are induced by several fundamental effects and can be classified in several ways [17][18][19] as shown in Figure 2.3. The discussion of the variations in the thesis is based on this classifications:

Spatial Variations: The important factor for this type of variation is from manufacturing process (sometimes also called as Process variations). Process variations are caused by the inability to precisely control the fabrication process for the nanometer technology. It is a combination of systematic effects(e.g., lithographic lens aberrations)and random effects (e.g., dopant density fluctuations). Examples of spatial variations include random dopant fluctuation, sub-wavelength lithography induced variations [17][19] etc. Random dopant fluctuations also impact the threshold voltage of the device [19].

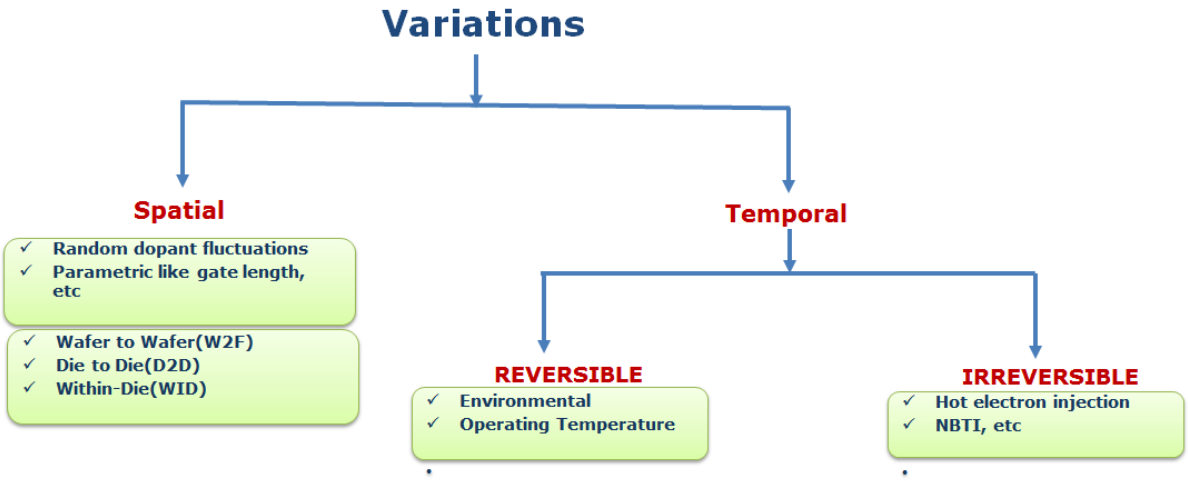

Figure 2.3: classification of variations derived from [3]

Temporal Variations: This variation is caused by the device operation over time. Temporal variations may be classified further as *reversible* or *irreversible*. Environmental variations constitute *reversible* variations while variations brought about by transistor wear-out or aging mechanisms, such as NBTI (Negative Bias Temperature Instability) and HCI (Hot Carrier Injection) [17][19][11] are a part of irreversible variations. NBTI manifests itself by degradation of PMOS threshold voltage [12][20]. The above effects gradually degrade process performance, albeit slowly during its operational lifetime. Voltage variation caused by the IR drops in the distribution network or by L dI/dt noise under changing load, variations constitute *reversible*. Local temperature hot-spots which is mainly due to different levels of activity across the processor also fall in this category.

In addition to these above PVT variations, input vector dependence of circuit delay is another major source of delay variations. These variations are sometimes difficult to capture unless we go for some design changes. Circuits exhibits worst case delay (also called as **Critical Path Delay**) for a particular instruction and data sequences [21]. Under adverse ambient conditions it is most likely that the circuits would fail, as it would not be operating in critical path all the time. Hence for most designs going for, a **worst-case** design with guard banding which is a traditional design approach is totally inefficient and not required. This effects the conservative design margins where energy wastage is not recommended. Even the Process variations have resulted in an increasingly lower yield if not taken care at design stage by means such as design for manufacturability (DFM) [22]. Therefore, it becomes increasingly imperative to address these issues in chip design.

2.3. TRADITIONAL DESIGN APPROACHES

The conventional design methodologies consider the worst case on timing margin, and try to design the circuit with timing yield high. However, unfortunately, the timing margin becomes greater and greater as the advanced technologies increase the variability. More the timing margin greater the power consumption. Hence, it becomes very difficult to satisfy both high yield and low power consumption under the worst-case design methodologies. To handle such variation due to process and circuit degradation problem in nanometer

technology, designers use the following traditional approaches as follows:

- Designers generally resort to corner analysis and design the circuit such that it is guaranteed to perform in the worst case scenario. Two common approaches are used one is a conservative supply voltage such that process variation degraded performance can still meet specifications [20]. These approaches are able to provide a guard band against projected process variations. However, they inevitably increase circuit power dissipation and therefore the designer hits another wall of nanometer integrated circuits *the increasingly tight power constraint*.
- Second approach is the designing the circuits with Over-sized transistors which usually imply unnecessarily large timing slack and therefore wasteful power dissipation.
- Alternatively, architectural approaches [12][11] are suggested for mitigating the variation problem. One technique is architectural-level adaptation [12][11] such as DVS (Dynamic Voltage Scaling) which is an improvement over setting a permanently high supply voltage. For instance, a chip can operate at relatively low supply voltage level when new and switch to higher supply voltage level when it detects process variation induced errors. Such adaptation can avoid the wasteful power. However, the DVS is a coarse grain technique, and the supply voltage level is usually fixed for major partitions of the chip, if not across the entire chip. In general, the variations and their effects vary among different components of a circuit. In order to ensure the performance of an entire chip, the DVS must be performed according to the worst process variation margin.
- That is, even though only 1% transistors may be strongly affected due to variations, the chip-level supply voltage has to be increased although the other 99% transistors have suffered very minor variations.

2.4. ADAPTIVE DESIGN APPROACHES

The traditional design approaches resulted in wastage of power and area, hence this has led to a significant interest in a new approach to chip design called **Adaptive technique**. The key idea of this approach is to tune system parameters (such as supply voltage and frequency of operation) during dynamic operation of processor. By dynamically tuning system parameters, such techniques mitigate the performance and power overheads of excessive margining. Thus, if the transistors are inherently faster, then the die automatically detects this and adjusts system parameters accordingly. Of course, voltage and frequency scaling needs to be within safe limits; otherwise, the consequent slow-down of the transistors can result in timing failures. More about the Adaptive Design techniques approach is explained in chapter 4.

3

INTRODUCTION TO 28 NANO-METER UTBB-FDSOI TECHNOLOGY

3.1. WHY FDSOI

In Large scale integration (LSI) circuit technology power reduction has become an important criteria. Bulk-Si devices are now running into a number of fundamental physical limits. Among the problems are that the carrier mobility is decreasing due to impurity scattering, the gate tunneling current is increasing as the gate insulator becomes thinner, and the p-n junction leakage is increasing as the junction becomes shallower [23]. These trends make conventional scaling less and less feasible. As a result, the operating voltage tends to be set higher than what a scaled-down device was expected to need to achieve the desired speed performance. The purpose of this chapter is to provide the reader with basic information on Fully Depleted Silicon-on-insulator (FDSOI) MOSFETs. The topics include the structure of an FDSOI substrate and the differences between Bulk and FDSOI MOSFETs .

3.2. WHAT IS FDSOI

The fig. 3.1 shows the structure of the FDSOI device [4]. FDSOI technology relies on an ultra-thin layer of silicon over a Buried Oxide (commonly called BOX). Transistors built into this top silicon layer are Ultra-Thin Body devices and have unique, extremely attractive characteristics. The top silicon layer is fully depleted, which means, it doesn't have any intrinsic charge carriers. The two flavors of buried oxide can be used:

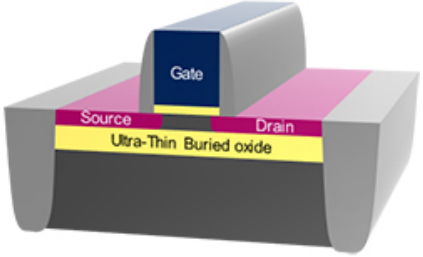

Figure 3.1: ultra-thin body and box (UTBB) FDSOI device

One is the Standard thickness (typically 145nm thick as classically in volume production FDSOI digital chips today), and the other is the ultra-thin BOX, for example 10 or 25nm (UTBOX, Ultra-Thin Buried Oxide)[4][23].

3.3. BULK VS FDSOI

Fig.3.2 (a)(b) illustrates a schematic comparison of bulk planar/silicon and UTBB-FDSOI device. Compared to bulk silicon UTBB-FDSOI provides lower junction leakage, near ideal SS, lower junction capacitance, high body biasing efficiency, lower DIBL and full dielectric isolation of the transistor. The low junction capacitance is importance for ultra-low power digital circuit switching power dissipation. Another main advantage of UTBB-FDSOI is the *body bias technique*; hence transistor can be controlled through two independent gates, called **Front gate** and **Back gate**, allowing dynamically modulating the transistor threshold voltage. Thus UTBB-FDSOI provides better transistor electrostatics [4] [24][23] .

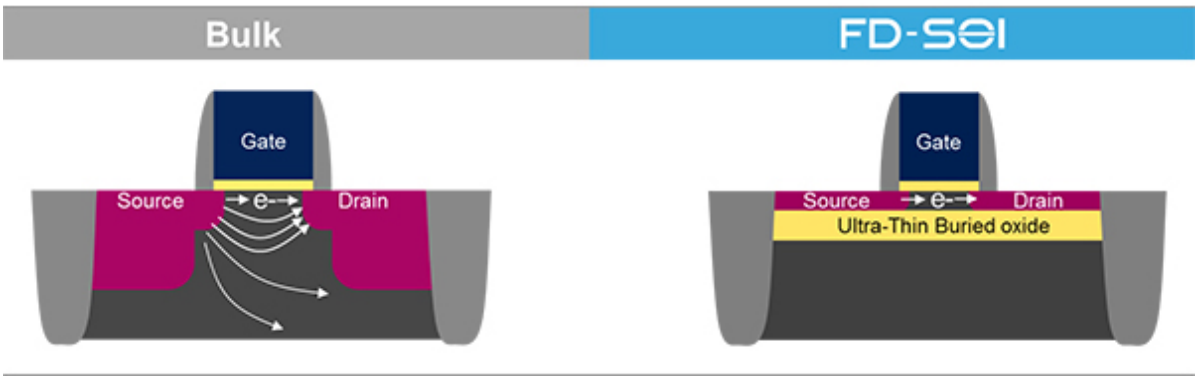

Figure 3.2: (a) Physical structure of bulk silicon, (b) ultra-thin body and box (UTBB) FDSOI device

Conversely, a Bulk approach for next generation technologies is expected to require more and poorer trade-offs in terms of performance vs. static and dynamic power consumption, yield vs. cell area, process complexity vs. leakage, etc. Virtually all low power techniques currently employed in classical Bulk CMOS technology can be directly ported to FDSOI [4]. One special case is Body Biasing, which can be very efficiently adapted to FDSOI in the form of back-plane biasing, using ultra-thin BOx wafers. Besides, voltage adjustments are no longer done via doping adjustments as FDSOI requires no channel doping (which is very advantageous for other reasons): alternative methods exist and are being assessed, including gate stack engineering (same metals as on Bulk and less deviation from mid-gap), back-plane biasing (with ultra-thin BOx) and VDD adjustment.

3.4. ADVANTAGES OF FDSOI

FDSOI solves, with less process complexity, scaling, leakage and variability issues to further shrink CMOS technology beyond 28nm. FDSOI offers the following major benefits [4] [24][23]:

- The excellent electrostatic control of the transistor, intrinsic to FDSOI, acts as a performance booster and enables lower VDD (therefore lower power consumption) whilst reaching remarkable performance.

- FDSOI is intrinsically Low Leakage and regains good control of Short Channel Effects. One consequence is the ability to aggressively shrink the gate length, making it easier to fit devices into smaller and smaller pitches and therefore increase logic density to continue Moores law [8].
- In addition, FDSOI transistors (which require no halo/pocket implant) natively offer superior analog behavior. This comes with other classical advantages of SOI like much improved Soft-Error Rate, etc.

3.5. UTTB-FDSOI DEVICE TYPES:

There are two standard V_{th} devices:

1. 1.0 V low V_{th} transistors (LVT) seated on flip-Wells enable to apply high forward back-biasing (FBB) up to 3V.
2. 1.0 V regular V_{th} transistors (RVT) built on classical-Wells; enable strong reverse back-biasing (RBB) down to -3V.

In our thesis proposal we are going to use LVT structure for all simulations as it has a V_{th} of 400 mV compared to RVT (V_{th}= 480 mV) [4]. This low V_{th} makes the LVT structure *FBB oriented*. As LVT is fabricated using the flip-well structure when compared to a standard well structure as shown in figure 3.3 . Thus with NMOS having N-well (NW) and PMOS with P-well(PW) the parasitic diode is forward biased when the bulk voltage (V_{bp})of PMOS is greater then the bulk voltage (V_{bn}) of NMOS. Hence this causes large leakage current, So in order to avoid this parasitic leakage current the condition $V_{bp} \leq V_{bn}$ should always be met to keep the diode in reverse bias mode (see fig. 3.3).

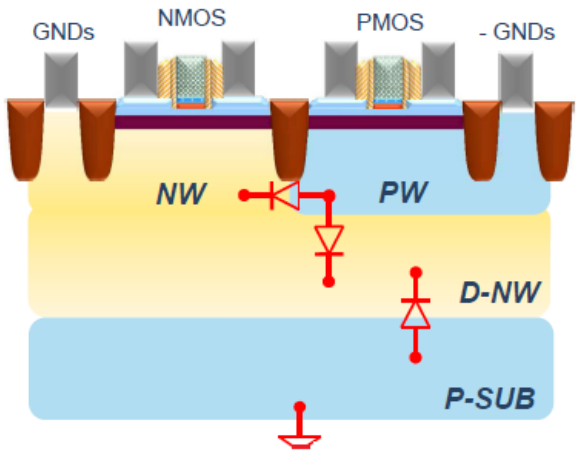

Figure 3.3: 1.0 V Low V_{th} transistor (LVT) structure from [4]

3.6. 28UTBB-FDSOI: BODY-BIASING

The transistor threshold voltage (V_{th}) is the key process parameter which is subject to variation due to many factors as discussed in the variability problem chapter. Variation in V_{th} directly impacts two major properties of the processor, namely the frequency it attains and

the leakage power it dissipates. Moreover, V_{th} is also a function of temperature, which increases its variability [1][25].

A body bias (BB) is a voltage applied between the source or drain of a transistor and its substrate, effectively changing the transistors V_{th} [1][25]. Depending on the polarity of the voltage applied, V_{th} increases or decreases. If it increases, the transistor becomes less leaky and slower; if it decreases, the transistor becomes leakier and faster. By reducing the V_{th} in cells with slow transistors and increasing the V_{th} in cells of leaky transistors, we reduce the variation within the die and attain a better frequency-leakage operation for the chip.

The BB range in the bulk technology is limited to -300mV in RBB due to gate induced drain leakage (GIDL), while FBB is limited to +300mV because of source-drain junction leakage and latch-up risk at higher voltage and temperature. The UTBB technology enables an extended body bias range from -300 mV (RBB) to +1.3 V (FBB)(proven voltages). This provides designers with a new lever for energy-efficiency optimization, performance boosting, ultra-low-voltage functionality and leakage reduction. Depending on the polarity of the voltage applied it can be categorized as:

1. **Forward Body Bias (FBB)** :In Forward BB (FBB), the voltage polarity is such that V_{th} decreases, creating a faster and leakier transistor ($V_{bn} >$ gnd , $V_{bp} \leq$ gnd).

When a positive gate to source voltage (V_{GS}) is applied to NMOS with proper biasing conditions and when V_{GS} is sufficiently large ,then the large number of electrons are attracted under the gate. This layer formed under the gate is called as *inversion layer*, and the repelled holes below the inversion layer creates a region which is a *deletion* region as its depleted of mobile electrons. The opposite is true for the PMOS structure.

By applying a positive voltage across the source to substrate (V_{bn}) for NMOS reduces the threshold voltage. When the FDSOI is forward body biased, the width of the depletion region beneath the gate decreases. Reducing the depletion width corresponds to decrease in the ionic charge on the capacitor plate formed between the gate and the substrate (bulk). In order to maintain the charge balance, the mobile charge in inversion layer which are electrons in case of NMOS are increased. As a result of increased mobile charge carriers, the gate voltage needs to achieve the similar level of charge balance compared to inversion layer, hence the threshold voltage for FBB FDSOI decreases. Similarly the opposite is true for PMOS structure [1][26].

4

STATE OF THE ART- TYPICAL CASE DESIGN

4.1. INTRODUCTION

As the traditional approach which is explained in the chapter 2 section 2.3 will not work. Considering this situation, design methodology should be reconsidered. Typical-case design methodology is one of the promising ones. It exploits an observation that worst cases are rare. Designers should focus on typical cases rather than worst cases. Since they do not have to consider worst cases, design constraints are relieved, resulting in easy designs. In the typical-case design methodology, designers adopt two methods to a circuit design at a time [22].

- **Performance-oriented design:**In this method only typical cases are under consideration. Since worst cases are not considered, design constraints are relaxed, resulting in easy designs.
- **Function-guaranteed design:**In this method worst cases are considered, designers dont have to consider performance. They only have to guarantee functions, and thus design must be simple, resulting in easy verifications.

Every critical function in an LSI chip is designed by two methods. The design consists of two components as shown in Figure 4.1. One is called **Main part** , and the other is called **Checker part**. While two parts shares the single function, their roles and implementations are mutually different. On designing the *main part*, performance is optimized to increase, but correct function is ignored to guarantee. The *main part* might cause errors. That is, it is implemented by the performance-oriented design.

The *checker part* is provided as a safety net for the unreliable main part. It detects errors that occur in the main part, and thus it has to satisfy all design constrains in the chip. However on the checker part design, while designers have to guarantee the function, they do not have to optimize neither of performance and power. That is, it is implemented by the function-guaranteed design. If an error is detected by the *checker part*, the circuit state has to be recovered to a safe point where the error is detected by any means.

Examples of the typical-case designs include Razor [3][21], approximation circuits [27], ANT [28]. Some of the methods are explained in the below section adaptive techniques state of the art.

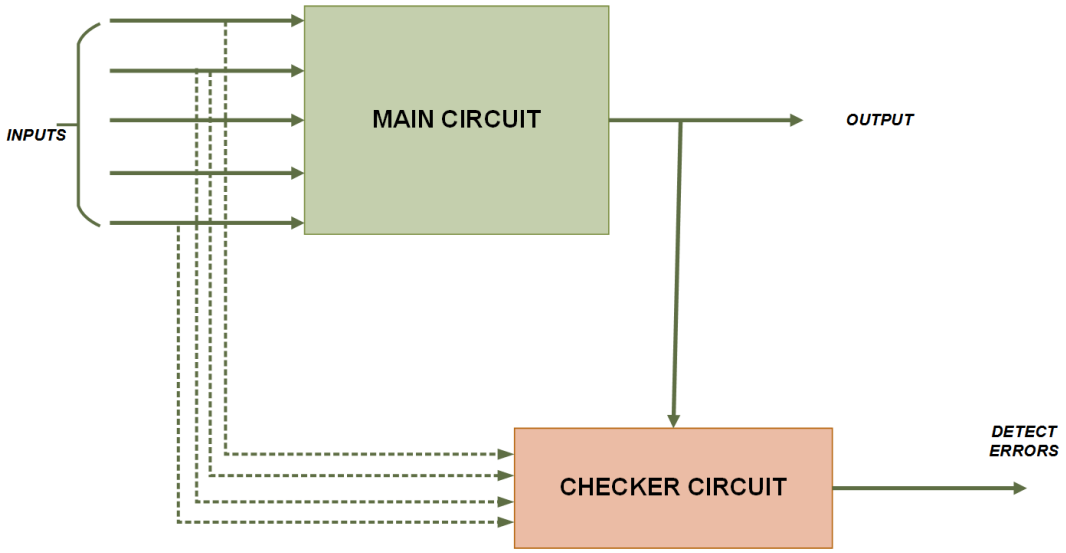

Figure 4.1: Typical Case Design

4.2. ADAPTIVE DESIGN TECHNIQUES

Adaptive techniques tune system parameters based on variations occurring in silicon dice and surrounding ambient environment conditions. In traditional methods either the voltage or frequency for all dies was tweaked in-order to increase the efficiency. Hence, in adaptive technique instead of using single operating voltage and frequency point for all dies, the system adjusts the parameters to deliver the better energy-efficiency. As mentioned in the introduction section, the different technique adopted for changing the voltage and frequency is discussed in detail. From [29], the adaptive technique can be broadly classified into two main categories as:

1. **Always correct**

2. **Let fail and correct**

Table 4.1, lists the different adaptive architectures [29][30][29][28], we are going to touch a few of them listed from the table 4.1. The key idea in the **always correct** is to predict the operational point where the critical-path fails to meet timing and to guarantee correctness. This is done by adding a safety margin to the timing parameter so that it predicts the failure before it happens. The conventional approach of predicting this failure point is to use either *look-up table* or *canary circuits*(This is similar as Circuit failure prediction).

Similarly the key idea for **Let fail and correct** is to scale the system parameters (e.g. voltage and frequency) till the point where the processor fails to meet timing, thereby leading to an error. An error-detection block flags the occurrence of the timing error upon which a recovery infrastructure is engaged to achieve correct state. The disadvantage is that an additional controller is required for the error correction mechanism. From the table 4.1 the Razor, Algorithmic Noise Tolerance(ANT),etc. [30][29][28] uses this technique (This is similar as Error detection).

Table 4.1: Different Adaptive techniques scenario

Category	Adaptive technique	Data error correction
21 **Always-correct or Circuit failure prediction**	Table-Lookup	No
	Canary circuits	No
	Typical-Delay Adder circuits	Yes
Let fail and correct or Error detection	Self-calibrating interconnects	Yes
	ANT	Yes
	Razor circuits	Yes

4.3. ALWAYS CORRECT

4.3.1. LOOK-UP TABLE BASED APPROACH

In the look-up table based approach [29], the processor is pre-characterized during design-time to obtain its maximum obtainable frequency for a given supply voltage. The safe voltage-frequency pairs are obtained by performing conventional timing analysis on the processor. Typically, the operating frequency is decided based on the deadline under which a given computational task needs to be completed. Accordingly, the supply voltage corresponding to the frequency requirement is *dialed in*.

4.3.2. CANARY CIRCUIT

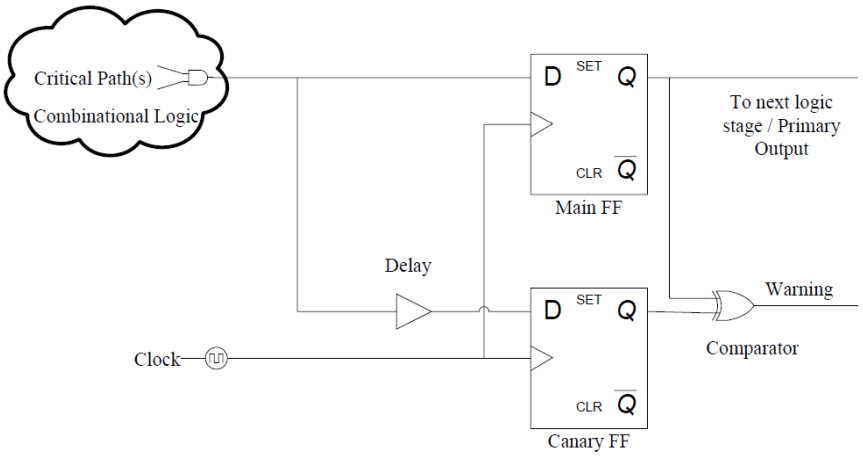

Figure 4.2: Canary Logic

The canary circuit [22] is for detecting variation-induced performance degradation in a predictive manner. The canary FF [31] is augmented with a delay element and the shadow FF, as shown in Figure 4.2. The canary FF acts like a canary in a coal mine which detects whether an timing error is about to occur. As shown in Figure 4.2, a canary circuit consists of two flip-flops; a main FF and a canary FF. The main FF gets the direct input and the ca-

nary FF which serves as the checker part gets the input through a delay buffer. This delay in the input reaching the two flops serves as the guard band for error detection. The outputs from these flops are fed to a XOR gate which functions as a comparator, outputting 1 when these are different and thereby predicting the occurrence of an error. Some advanced designs of canary circuits are proposed in [20][19]. More detailed explanation of the canary logic operation with timing diagram is explained in chapter 5.1.

4.4. LET FAIL AND CORRECT

4.4.1. SELF-CALIBRATING INTERCONNECTS

Self-calibrating interconnects [5] Fig. 4.3 address the problem of reliable on-chip communication in aggressively scaled technologies. Signal integrity concerns require on chip buses to be strongly buffered which consumes a significant portion of the total chip power. Hence, it is desirable to transfer bits at the lowest possible operating voltage while still guaranteeing the required performance and the targeted bit-error-rate(BER).

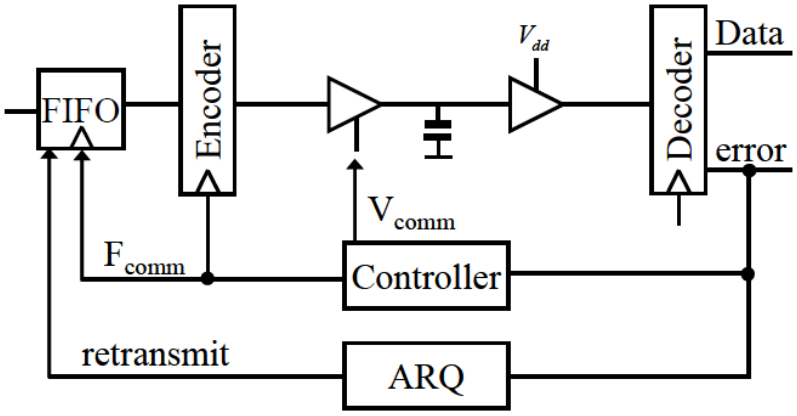

Figure 4.3: Self-calibrating interconnects from [5]

The receiver is augmented with a checker unit that decodes the received code word and flags timing errors. Correction occurs by requesting retransmission through an Automatic Repeat Request (ARQ) block, as shown in Figure 4.3. Furthermore, an additional controller obtains feedback from the checker and accordingly adjusts the voltage and the frequency of the transmission. By reacting to the error-rates, the controller is able to adapt to the operating conditions and thus eliminate worst-case safety margins.

4.4.2. RAZOR

Razor relies on the combination of architectural and circuit techniques to achieve efficient error detection and correction of timing violations. Razor [3][12] permits to violate timing constraints to improve energy efficiency. Razor works at higher clock frequency than that

determined by the critical path delay. In order to detect timing errors, Razor flip-flop (FF) shown in Figure 4.4 is proposed.

Each timing critical FF (main FF) has its shadow FF, where a delayed clock is delivered to meet timing constrains. In other words, the shadow FFs are expected to always hold correct values. If the values latched in the main and shadow FFs do not match, a timing error is detected. When the timing error is detected in microprocessor pipelines, the processor state is recovered to a safe point with the help of a mechanism based on counterflow pipelining.

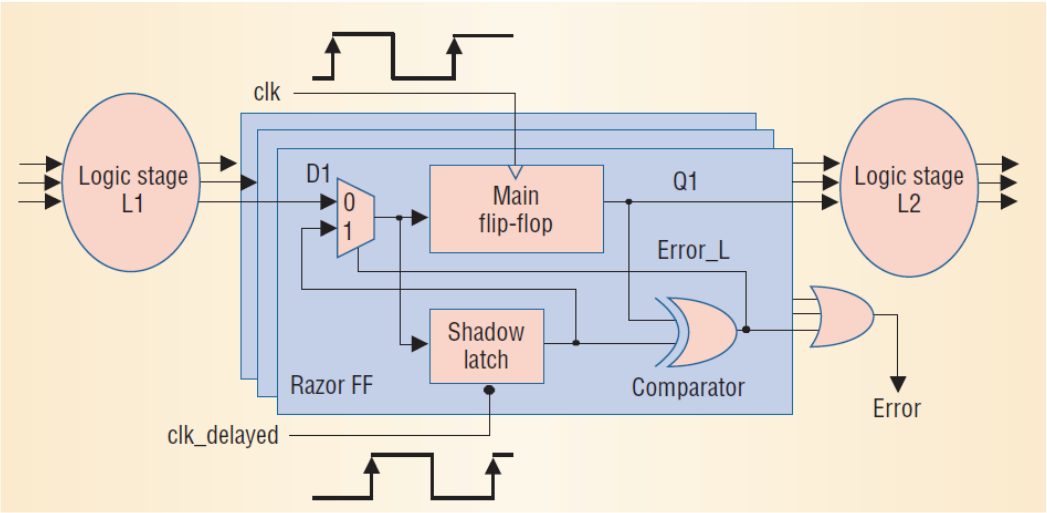

Figure 4.4: Razor Flip-Flop

Razor removes voltage margin for power reduction. The voltage control adapts the supply voltage based on timing error rates. Figure 4.5 shows the Razor's dynamic voltage scaling system. If the error rate is low, it indicates that the supply voltage could be decreased. On the other hand, if the rate is high, it indicates that the supply voltage should be increased.

The control system works to maintain a predefined error rate, E_{ref}. At regular intervals the error rate, E_{sample}, is computed and the rate differential, $E_{diff} = E_{ref} - E_{sample}$, is calculated. If the differential is positive, it indicates that One of the difficulties on Razor is how it is guaranteed that the shadow FF could always latch correct values. The delayed clock has to be carefully designed considering so-called short path problem [21].

4.5. CANARY FF VS RAZOR FF

Razor is a combination of architectural and circuit technique which is an efficient way to detect and correct timing errors which eliminates design margin. The canary FFs has the following advantages when compared to Razor technique:

1. **Elimination of the delayed clock:**The important net in physical routing is the sensitive clock pin. The designers avoid using multiple clocks which are prone to noise,

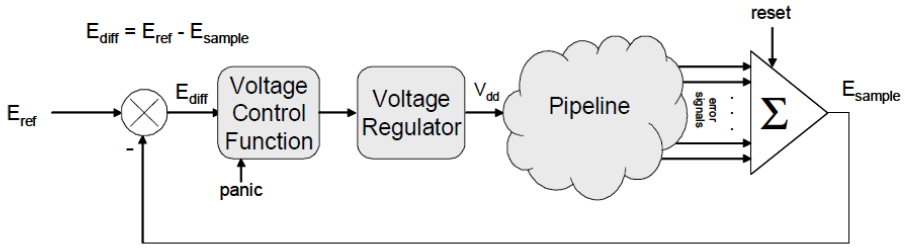

Figure 4.5: Razor's Voltage Scaling System

jitter, phase error,etc to simplify the clock tree routing during the design and routing phase. The delay buffer always has a positive delay, even though parameter variations affect it. Hence, the canary FF encounters a timing error before the main FF.

2. **Elimination of re-execution mechanism technique:**Razor technique requires a re-execution mechanism to correct timing errors. The re-execution is performed through architectural replay, which is often integrated in high-performance processors to support branch prediction [13]. However, it is impracticable for general sequential circuits and simple processors [5][32]. In contrast, canary FF predicts the occurrence of timing errors. This means that any error recovery mechanisms are not needed as long as the prediction is appropriate. Therefore, it is suitable to apply canary FF to small circuits or processors.

3. **Complicated buffer insertion:** Razor FF requires the timing window of error detection just after the clock edge in order to detect a late-arriving signal as a timing error. Thus signals arriving during the timing window are considered as timing errors. This means that the timing window is equivalent to a hold time of Razor FF.Consequently, the timing window, which is set to be large for capturing large setup-time violations, is much larger than a hold time of a normal FF and canary FF and hence Razor FF inherently suffers from severer minimum path delay constraints compared to a normal FF and canary FF. This makes design of the buffer-insertion more complicated

5

DESIGN APPROACH

5.1. INTRODUCTION

The goal behind this thesis proposal is to design a circuit which predict the occurrence of a failure during normal system operation before the appearance of any error that can result in corrupt system data and states. Failure prediction can be performed in many ways as it is discussed in previous chapter 4 about the state of the art system for error detection system. The basic principle behind these is to insert a number of circuit which work like a *sensors* at various locations inside a chip creating islands for the adaptive control. Examples of such system parameters range from temperature, voltage, ring oscillator delays to complex relative timing relationships among logic signals [15][2][16]. The data collected by the sensors is analyzed on-chip or off-chip to identify *anomalies* and predict failures. Sensor designs and data analysis techniques can widely vary depending on failure sources and their sensitivities [33][34][29]. The purpose of this thesis is an approach to circuit failure prediction for PVT variations by designing an adaptive body bias (BB) system using Canary logic [22].

When we consider the pros and cons of **Circuit failure prediction** and **Error detection**, unlike error detection a major limitation of failure prediction is that not all circuit failures can be predicted, for example radiation-induced soft errors. In circuit failure prediction there are no corrupt data as the errors are predicted before it happens. Conversely in the error detection we cannot avoid error states hence there is a latency problem compared to the later scheme. The error detection is generally expensive as there is need to implement the pipeline recovery mechanism [13]. In circuit failure prediction data is collected over several clock cycles, such data analysis can also be extremely beneficial for self-diagnostics. However the failure prediction cannot predict all failures as this will be discussed with Canary logic implemented in our design approach.

The major contributions of this thesis proposal are:

- Introduction of the concept of circuit failure prediction, and demonstration of its practicality and effectiveness for PVT variations.
- Design of special sensors to demonstration of their efficiency and effectiveness.
- Demonstration of new design techniques that enable close to best-case performance unlike traditional design techniques that impose worst-case speed margins.

- Avoid the use of DAC based conventional design [15][2][16] in order to reduce the area and to make the circuit less complicated.

This section is divided into two main parts: In first half an overview towards implementing adaptive BB for digital circuits is presented and in second half, complete explanation with respect to its implementation.

5.2. MAIN SYSTEM LEVEL DESIGN

The proposed adaptive body bias system consists of the following main blocks as depicted in Figure 5.1:

1. Main Test Circuit (4-bit Adder and 8-bit Multiplier using full adder (FA) structure).
2. Canary logic block.
3. Control Unit block.
4. Reference Pulse Generator block.
5. Body Bias (BB) Generator block (Phase detector and Charge Pump).
6. Voltage control delay line (VCDL).

5.2.1. OVERVIEW

The idea proposed for adaptive system is shown in figure 5.1. The working of the proposed system is as follows:

The VCDL and Phase detector together works like a conventional delay locked loop (DLL) system. VCDL works like a delay sensor for the main test circuit, using a target delay a timing pulse generated from reference block (REF PULSE). The Canary circuits predicts the occurrence of failure in a circuit which results in a corrupt data and states. The *Warning signal* issued from the Canary circuits indicates that the main circuit is too slow and FBB must be applied to increase the speed. The occurrence of warning signal is counted by an UP-DOWN counter implemented along with control logic and a timer unit which asserts a timer monitor flag when the monitoring period of the warning signal is elapsed. The occurrence and lack of warning signal is used to control the width of REF PULSE and therefore the target delay of the VCDL (which is linked to the main circuit critical path). The output of VCDL adjusts the value of FBB until the target delay defined by the REF PULSE is achieved.

There are several implementations to control the circuit speed, such as supply voltage scaling and body-biasing. Since several works have been pointed out for the dynamic V_{th} scaling(DVTS) [35], Variable and Dynamic supply voltage scheme(VS) [36][37] but the adaptive body-biasing technique is more efficient for low power operation and sub-threshold circuits [38][31]. For the adaptive speed control, multi-level body-bias voltages are required. In many of the previous designs the body bias is applied using methods like selecting fixed voltage [33][34], D/A converter , etc. [15][2][16].

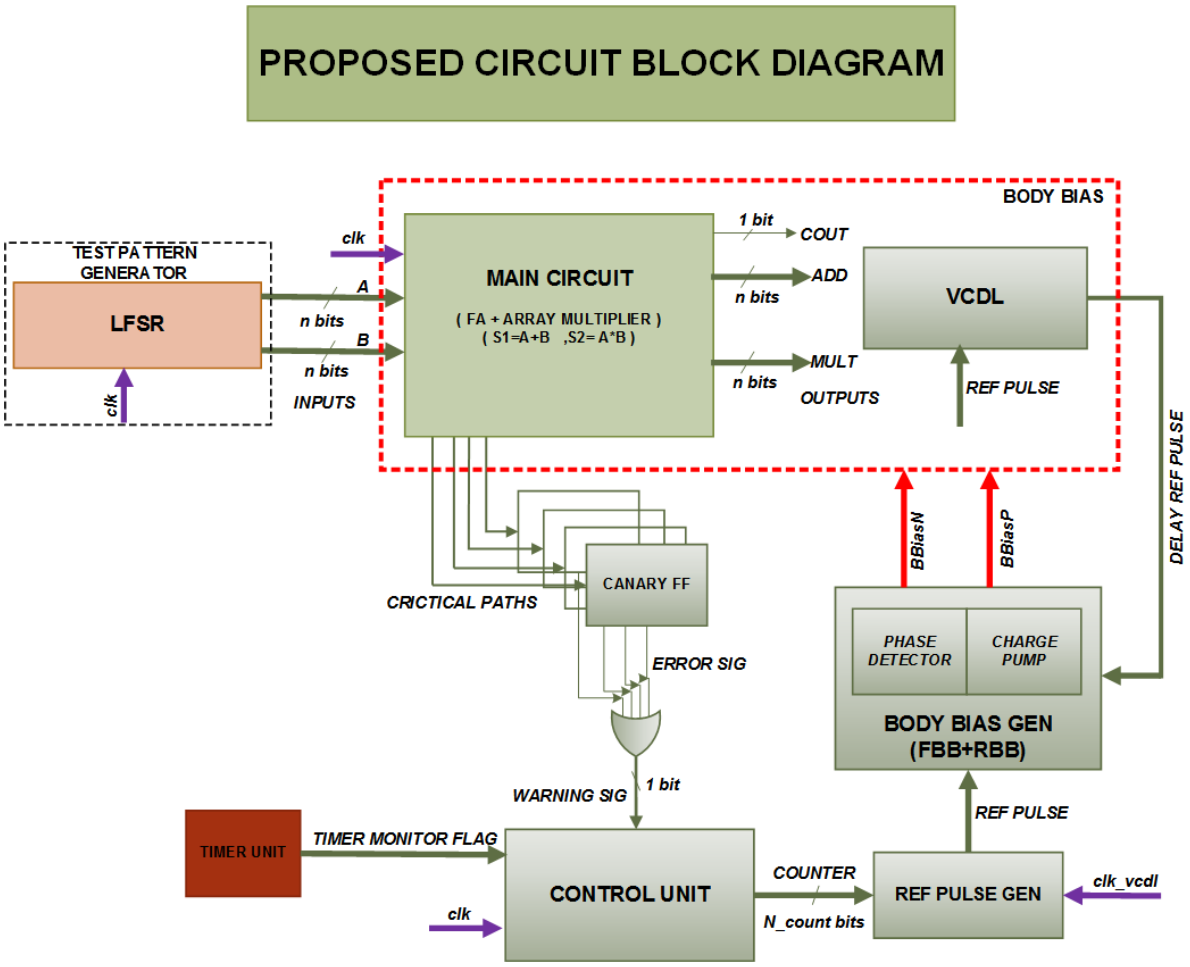

Figure 5.1: Proposed block diagram: Adaptive body Bias system

On the other hand, the adaptive speed control with canary FF presented makes the circuit speed slower to reduce the power dissipation. In this case, timing errors cannot be completely eliminated when the adaptive speed control is applied to normal operations. This is because the circuit might be slowed down excessively when the paths, where canary FFs are inserted [39][40][20].

5.3. TEST CIRCUIT IMPLEMENTATION- ADDER AND MULTIPLIER

In order to show the adaptive body bias operation, we have implemented most fundamental arithmetic, a simple adder and multiplier. The adder sums two 3 bit input and output is a 4 bit with one bit carryout(C_{out}). Similarly, we have a 8 bit output multiplier with each 4-bit multiplier and multiplicand respectively as shown in the Fig 5.2. The hardware required for the generation of these partial products are basic gates (AND,OR,NAND,NOR). Using any adder like Carry Save Adder (CSA), Carry Propagate Adder (CPA) we can add the partial products. In this method we are using simple full adder (FA) and 4X4 array multiplier for simplification.

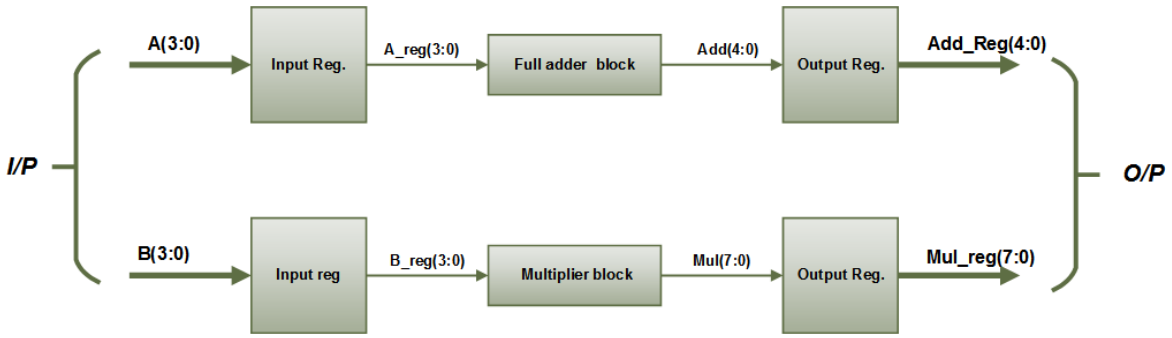

Figure 5.2: Proposed Test circuit: 4-bit adder and 8-bit multiplier

In the array multiplier method, the first row will be either Half-Adders or Full-Adders. If the first row of the partial products is implemented with Full-Adders,C_{in} will be considered *0*. Then the carries of each full adder can be forwarded to the next row of the adder (see Figure 5.3). An array multiplier uses short wires that go from one full adder to adjacent full adders horizontally, vertically. An n × n array of AND gates can compute all the a multiplied b terms simultaneously. The terms are summed by an array of $n \times (n - 2)$ full adders and (n) half adders. The number of rows in array multiplier denotes length of the multiplier and width of each row denotes width of multiplicand. The output of each row of adders acts as input to the next row of adders. In future section we discuss about the timing analysis performed on the adder and multiplier part.

5.4. CANARY LOGIC IMPLEMENTATION

CANARY LOGIC WORKING PRINCIPLE

The canary circuit [22] is for detecting variation-induced performance degradation in a predictive manner. As shown in Figure 5.4 a canary circuit consists of two flip-flops a main FF and a canary FF. The main FF gets the direct input and the canary FF which serves as the checker part gets the input through a delay buffer. This delay in the input reaching the two flops serves as the guard band for error detection. The outputs from these flops are fed

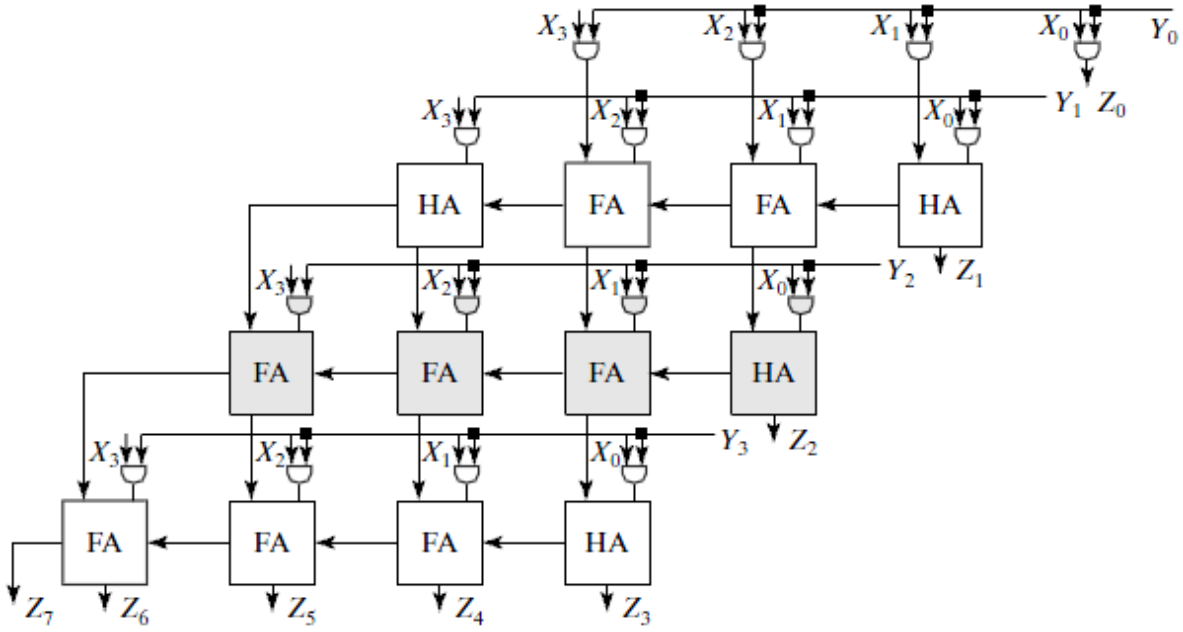

Figure 5.3: Conventional 4X4 array multiplier for unsigned numbers

to a XOR gate which functions as a comparator, outputting 1 when these are different and thereby predicting the occurrence of an error.

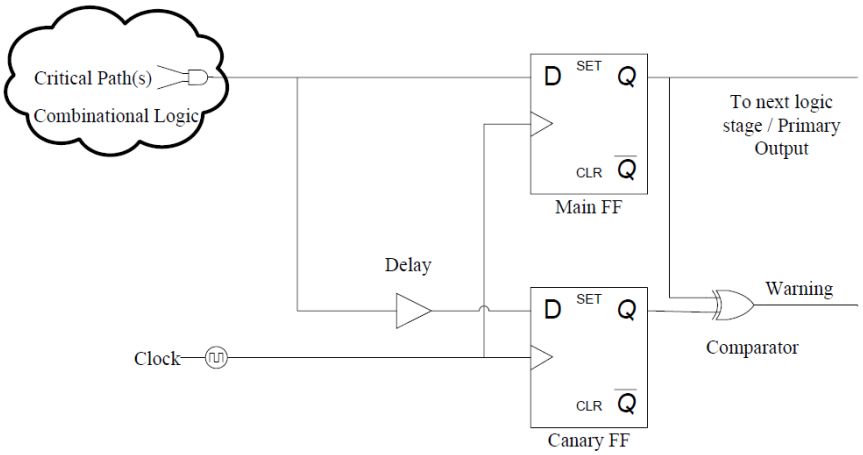

Figure 5.4: Canary logic Circuit

Canary circuit is a typical case design alternative of Razor [3]. However, in contrast to Razor, which delivers a delayed system clock to the checker part (shadow FF), canary circuit delivers a delayed input signal to the checker part (canary FF). This simplifies the clock tree synthesis and routing as there is just one system clock now. Also, the delay buffer placed before the canary flip-flop always has a positive delay.

Even if affected by process variation, which makes the canary flip-flop recover from variation induced effects by itself. Canary circuit also predicts timing errors rather than

detecting them afterwards. The predictive warning allows the user to take preventive measures before the timing violation actually occurs and thus the system does not run into any corrupt data states, except for errors that cannot be predicted such as single event upset (SEU) errors. However, timing violations caused by the variations of our interest can be predicted effectively by architectures such as canary circuit.

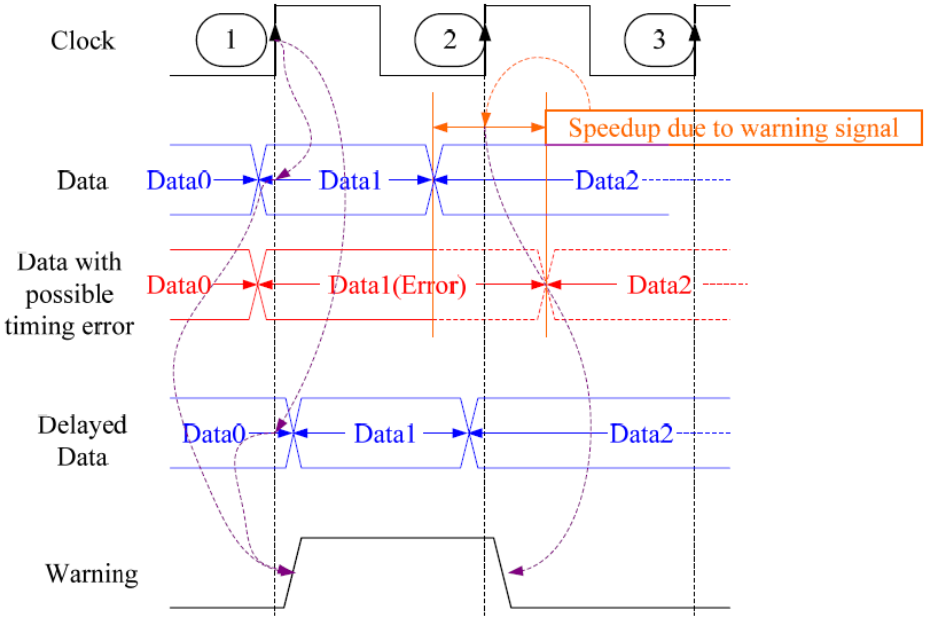

Figure 5.5: Error prediction and prevention using canary circuit

The waveform in figure 5.5 shows how the approach of this thesis uses the canary circuit to predict and prevent the occurrence of timing violations. Following the labels used in figure 5.5, Data is the input to the Main FF and Delayed Data is the input to the Canary FF. On the first clock edge, the main FF clocks in Data1 and the canary FF clocks in Data1 thereby causing a mismatch between their outputs and raising the Warning signal high. This indicates that the canary circuit has predicted a timing error.

As described in the overview of the Main circuit system level, this forward biases transistors on the critical paths. As a result Data1 is sped up through the combinational path and it arrives at the Main FF such that the timing requirements are satisfied at the second clock edge. Thus, the error is prevented from occurring. In the absence of any error prevention mechanism, the data at the main FF would not have been as shown in red color in figure 5.5.

As mentioned in the main test circuit implementation section, all the 8-bit output from the product multiplier are inserted with, *canary FF*. Hence each of the *canary FF* generates an *error signal* (in-case it detects any timing violations), this signal from the canary FF are *ORED* together to get the final **Warning signal** as shown in figure 5.6. This **Warning signal** is the one which is used as an input for the control circuit.

To implement the buffer delay for the proposed design the simulation is performed to

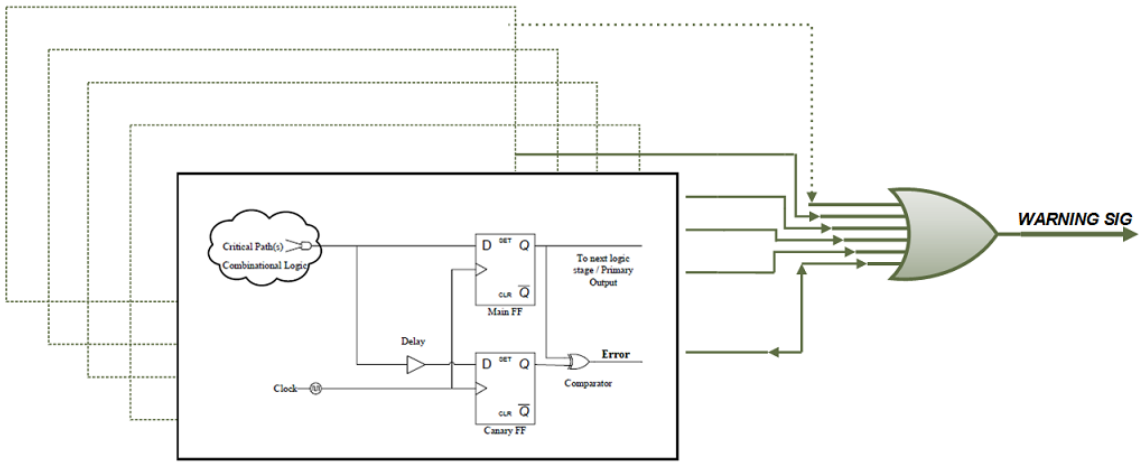

Figure 5.6: Canary Circuit: Error signal ORing to generate Warning signal

calculate the delay accordingly , the buffer delay required is **50 ps**. It is simply implemented by a *5-inverter chain*. In the next section we will discuss about the control unit section.

5.5. CONTROL UNIT

The control unit is the main section of the design which generates the increment and decrement signal which are used as an input to 2-bit UP-DOWN counter. The control circuit implementation is as shown in figure 5.7. There are two input signals: **Warning signal** is from the canary FF and another is **Timer monitor flag** generated from the monitoring unit respectively.

5.5.1. TIMER UNIT

The monitoring unit functions as a free running timer generating logic high pulse signal for a defined duration (configurable by using select lines). The generated pulse has a period which is equal to the input clock frequency duration. It checks for the occurrence of **Warning signal** (from canary logic unit) i.e if the **Warning signal** is logic low within a certain configured duration then the monitoring unit generates **Timer monitor flag**. This means that there is no error predicted and the circuit may work more slowly in-order to save power consumption. In case **Warning signal** is logic high then circuit needs to speed up to avoid future errors. As the timer is configurable by using select lines and for our design its configured for a duration of **250 ns**. This is a simple timer implemented using the 11-bit counter logic (see appendix A).

5.5.2. CONTROL UNIT - FLOW CHART

The figure 5.8 shows the decision flow chart implemented for the control unit. The following steps explains the design functionality in detail:

1. When the test circuit is powered on , it checks whether the test circuit is in **sleep mode** i.e. **reset mode**, if the reset signal is still asserted then the circuit will stay in No body bias (NBB) mode.

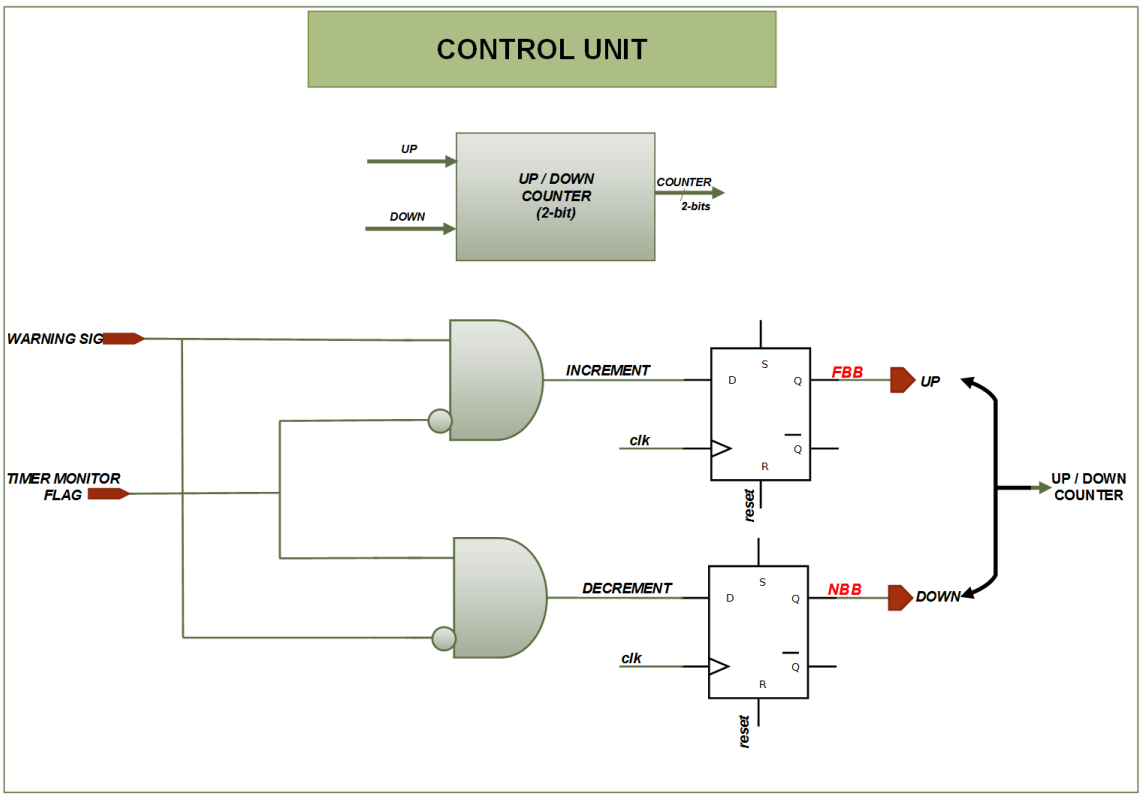

Figure 5.7: Proposed Control Unit

2. In the second step when the test circuit comes out of sleep mode to operating mode then it checks for the **Warning signal** from the *canary FF logic* circuit.

3. If the generated **Warning signal** is *logic high* then it asserts **INCR_flag** to *logic high* state else it will assert *logic low* signal to timer unit. (This INCR_flag signal is shown as UP signal in figure 5.7)

4. In the next step if the **Warning signal** is *logic low* then the timer unit monitors the asserted signal for configured time duration (in our design it is 250 ns). If there is no *logic high* **Warning signal** within the configured duration then the timer unit asserts the **Timer monitor flag** signal.

5. If the **Timer monitor flag** signal is *logic high* then it outputs the **DECR_flag = 1** else **DECR_flag = 0**.

As explained above in the control flow chart section, final output **UP** and **DOWN** signals are given to a 2-bit UP-DOWN counter as shown in the figure 5.7. From the flow chart (see figure 5.9) when the input signal **UP** is *logic high* and signal **DOWN** is *logic low* then the counter starts incrementing. When the counter value reaches a count value of '3' then counter stops counting and stays in that state or else if **UP** is *logic low* and **DOWN** is *logic high* then the counter starts decrementing until it reaches a count value of '0'. When the counter value is '0' then it stays in that state until it gets *logic high* **Up** signal.

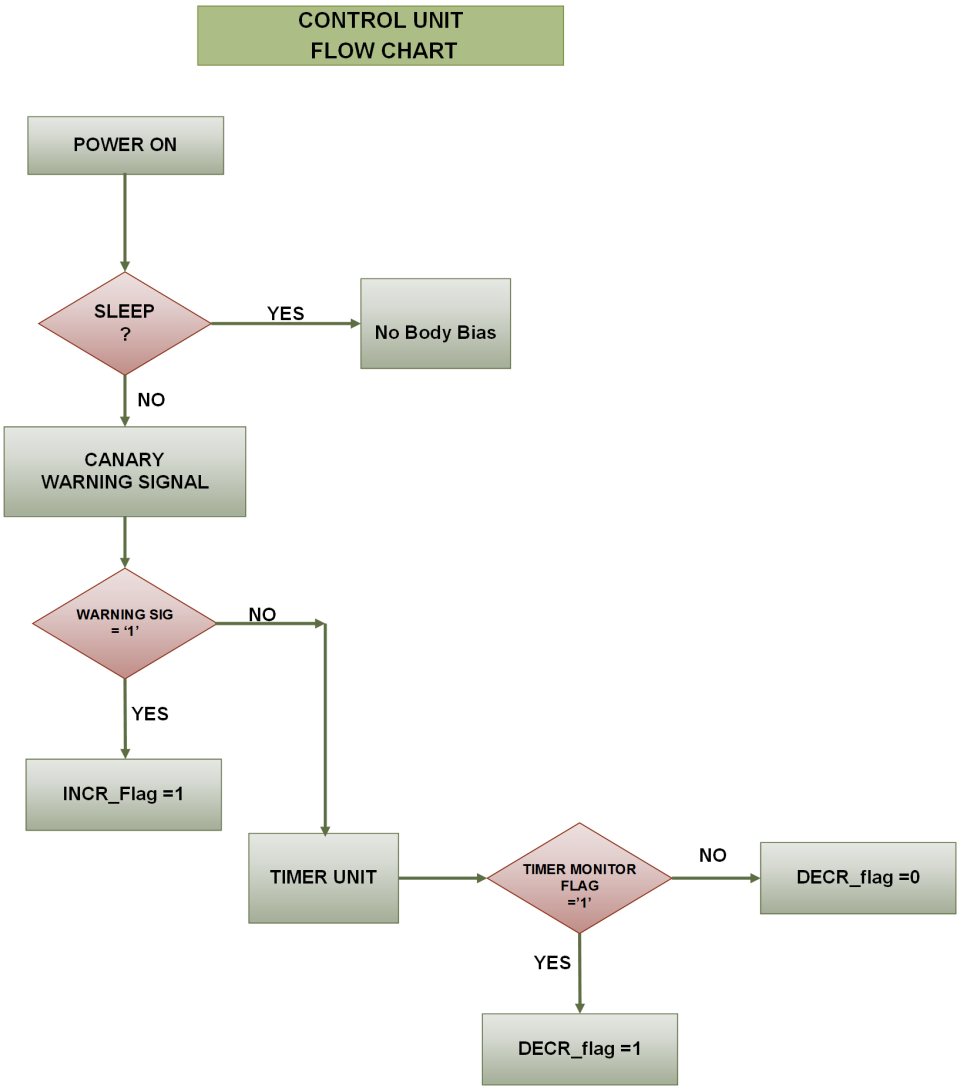

Figure 5.8: Control unit decision flow chart

5.6. REFERENCE PULSE GENERATOR

Pulse width modulation (PWM) is a powerful technique for controlling analog circuits with a processors digital outputs. PWM is employed in a wide variety of applications, ranging from measurement and communications to power control and conversion.

The main idea to use the DPWM pulse for the design proposal is to convert the digital count values to analog signal and to avoid the use of DAC based method used in conventional designs. This generated reference signal goes to the BBG block which uses the conventional DLL method to generate the required signal for the charge pump. Hence by this approach we try to avoid the use of DAC whose designing is more complicated and consumes considerable area.

The reference Pulse generator block is implemented using digital delay line method [6] which generates **reference pulse** signal depending on the counter values from the control unit section. This digital delay line method is also referred as *Digital Pulse Width Modulator(DPWM)*. The Figure 5.10 represents the basic delay line based DPWM circuit. In the

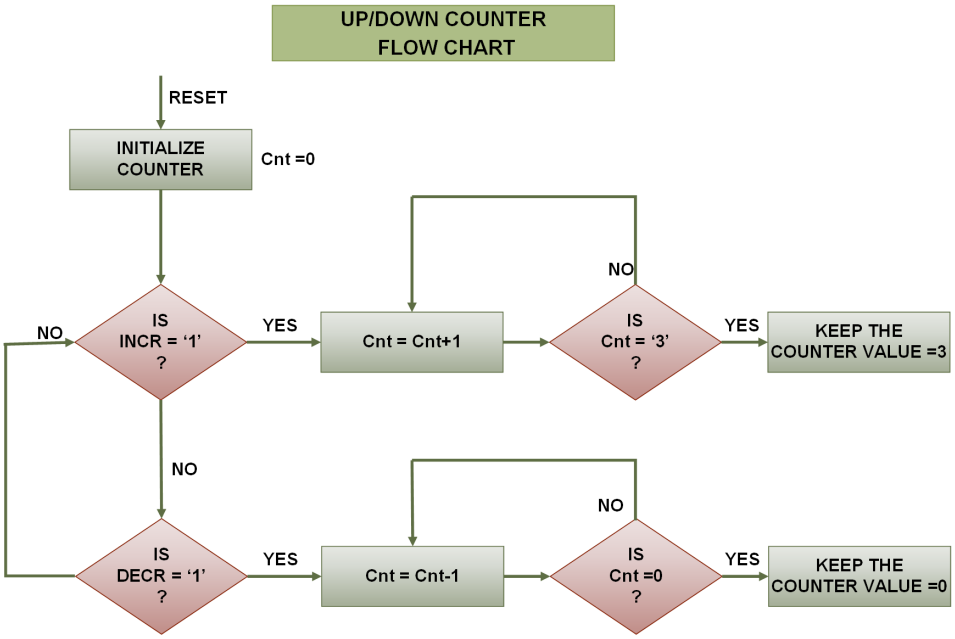

Figure 5.9: 2-bit UP-DOWN counter flow chart

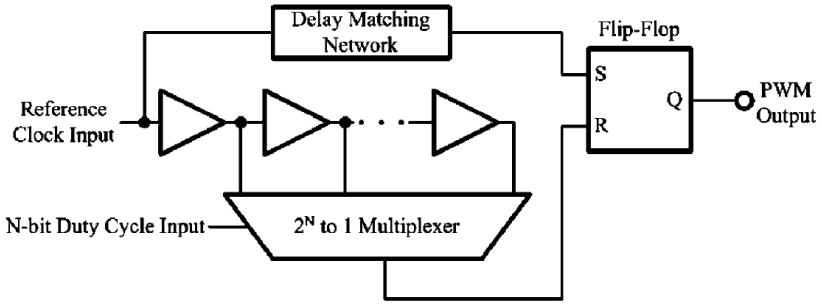

Figure 5.10: PWM generator: Delay Line based from [6]

delay-line-based PWM circuit a pulse from reference clock starts a cycle and after a certain delay, designed to match the propagation delay experienced through the multiplexer, sets the PWM output to high. The reference pulse propagates through the delay line and when it reaches the output selected by the multiplexer its value is used to set the PWM output to low. The drawback of this method is that the implementation area requirements grow exponentially with the multiplexer resolution bits **N**.

$$\Delta t_i = 4\Delta t_{i-1} = 2^{2i}\Delta t_0 \tag{5.1}$$

The Δt_i delays are created by simply replicating the Δt_0 delay cell, resulting in the same overall number of delay cells, neglecting the additional dummy load cells. As in the proposed design we are using a 2 bit counter, hence we can go for implementing the DPWM using both methods described above. But if the design is for higher counter values then using the improvised DPWM delay line reduces the delay elements required to implement it,

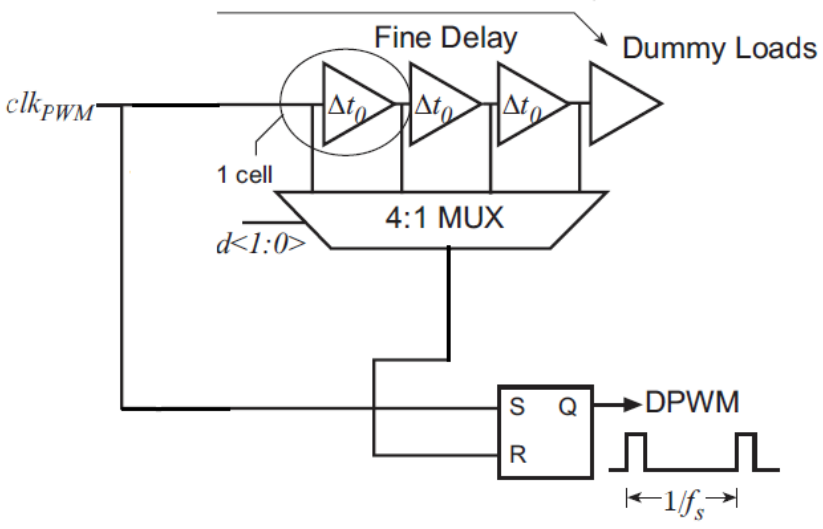

Figure 5.11: Alternate Simplified block diagram : A 1-segment, 2-bit DPWM used from [7]

hence improvisation in the transistor area. For example for a 6 bit DPWM, the multiplexer used changes from 64-to-1 multiplexer to 4-to-1.

In the design we are using a reference signal with a time period (clk_{vcdl}) 2 times the main operating clock ((clk) (see figure 5.1). By this we get double the duty cycle i.e ON period, as this ON period is used to generate discharge pulse by AND2 operation in the BBG part(see section 5.7). Hence the input frequency (clk_{vcdl}) for generating DPWM used is **600 ps**. The experiment results for the design is discussed in chapter 6. In the next section we will discuss about the body bias generator(BBG) which is used for generating the body bias voltage required for the adaptive control of the circuit.

5.7. BODY BIAS GENERATOR (BBG)

The body bias generator (BBG) consists of *Phase Detector* and *Charge Pump* along with *Voltage Controlled delay line(VCDL)* used for generating delay reference signal. BBG is used to generate forward body bias(FBB) voltages outside power supply rail. We have implemented a reset signal which acts like sleep mode to keep the circuit at No Body Bias (NBB)(($\mathbf{V_{bn} = 0\,V}$, $\mathbf{V_{bp} = 0\,V}$)) to reduce leakage current. Finally BBG works in a closed loop structure like a DLL architecture so that the timing variations caused by the main circuit is tracked by the proposed BBG by modifying the body bias output. This fully distributed architecture fashion makes this proposal very scalable and easily integrable into an automated design flow. Figure 5.12 shows the block diagram of implemented **BBG**.

5.8. VOLTAGE CONTROLLED DELAY LINE (VCDL)

The signal coming from an external pulse generator passes through a 24-stage Voltage Controlled Delay Line (VCDL) that, under zero body bias condition, will delay the reference pattern pulse according to PVT variations. The reference signal generated from the reference pulse generator has a varying pulse width depending on the multiplexer selection bit

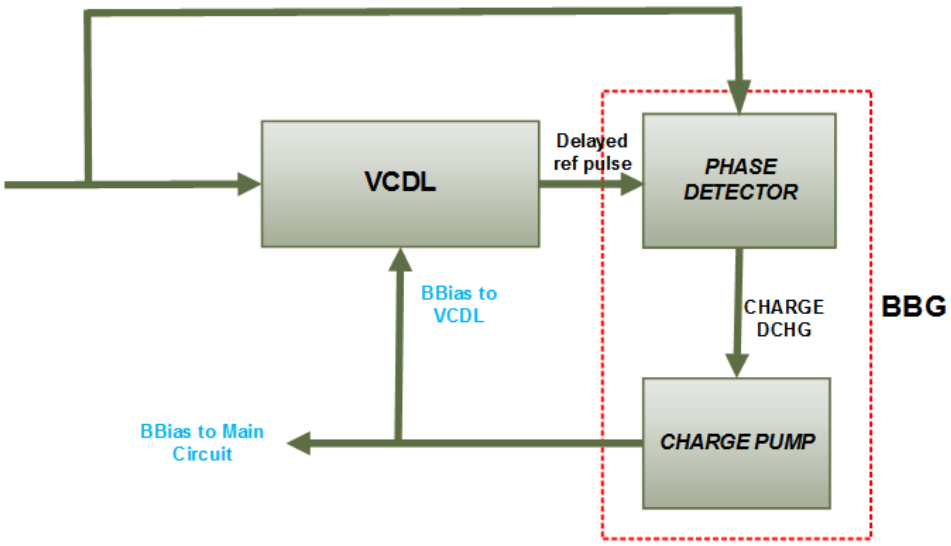

Figure 5.12: Block diagram: Body Bias Generator (BBG)

lines (S_1,S_0). Now the following steps tells how to fix the number of VCDL required which is performed under the absence of any variation for a given operating frequency.

1. First make sure whether the main test circuit is in reset (sleep mode) or operating mode.

2. If it is in reset mode then the initial value of the counter is *zero* and the circuit will be in NBB mode. The reference pulse width generated for this counter value, decides the initial number of VCDL required.

3. Once the initial number of required VCDL for attaining the end-end delay is calculated, then make sure (at this above condition mentioned in step 2) the rising and falling edge of the *reference pulse* and *Delay reference pulse* is between 10-90% of rise and fall time (see figure 5.13). This is done to keep the circuit under NBB mode when the counter value is at '0' value. By following the 3 steps we can decide the VCDL number required for the design.

The delayed and original reference pulse are fed to the phase detector circuit as shown in figure 5.12. The reference pulse frequency is set based on the desired BBG response time such that a faster response time implies a higher reference signal frequency at the expense of power consumption overhead. In next subsection, we will see the working of the phase detector and the charge pump based on the VCDL output.

5.9. PHASE DETECTOR

Phase detector compares the phase difference between two input signal and produce Output signal in form of difference voltage proportional to phase difference. In this design the phase detector compares the phase of input or reference signal with the phase of signal produced by the VCDL. The figure 5.14 is the phase detector block diagram used [41].

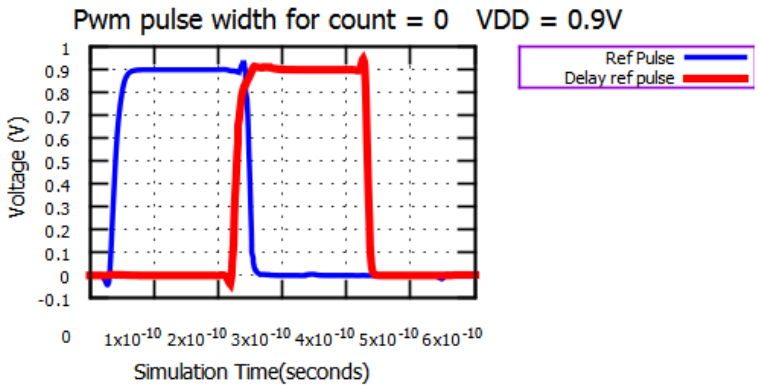

Figure 5.13: DPWM waveform for counter value '0'

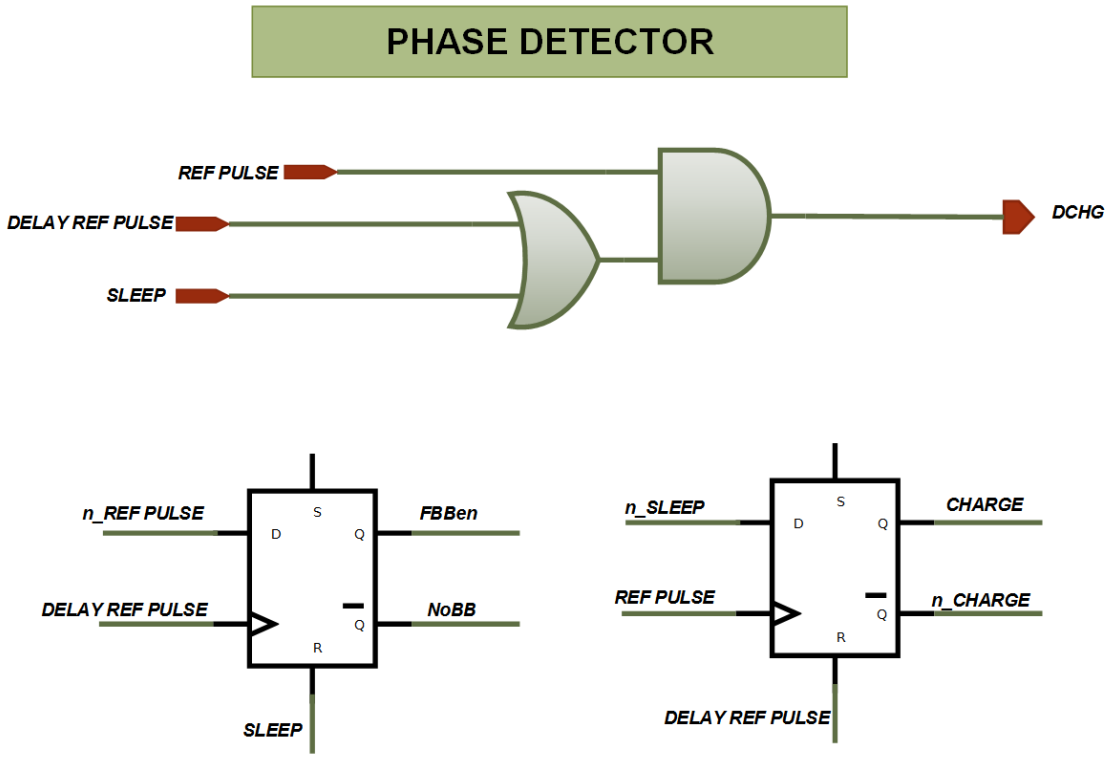

Figure 5.14: Proposed Phase Detector

Once the reference signal passes through the VCDL, both, delayed and reference signal arrive to the phase detector. The objective of FBB control is that (see figure 5.14) the VCDL output rising edge is produced at the same time as the reference signal falling edge, and thus VCDL nominal end-to-end delay (corresponding to the reference pulse width) see figure 5.15, is achieved. The phase detector generates *charge* and *discharge* pulses as well as two signals that determine the BBG operation mode (NBB_{en} or FBB_{en}). The first flip-flop shown in figure 5.14 determines the operation mode: the reference pulse logic level is latched after each rising edge of the delayed pulse. If the VCDL is too fast, the inverse of the

reference pulse will still be at low level, and therefore, NBB_{en} will be active, indicating that FBB must be decreased. NBB_{en} will also be active in sleep mode in order to save leakage power.

The second flip-flop in (Figure 5.14) generates a charge (CHARGE) pulse (only when reset or sleep mode is disabled) the width of which is proportional to the time difference between the falling edge of the reference pulse and the rising edge of the delayed reference pulse. Therefore, the slower the delay line, the wider the charge pulse. Finally, the discharge (DCHG) pulse is generated by making AND2 operation between the reference pulse and its delayed version in third (see figure 5.14). The discharge pulse width will increase with VCDL speed, and hence charge pump will generate NBB. Also in reset or sleep mode the phase detector will generate a discharge pulse.

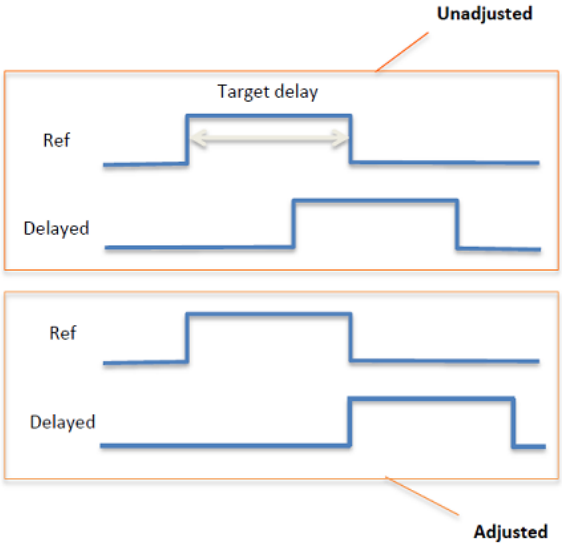

Figure 5.15: Timing diagram : Ref and Delayed pulse

5.10. CHARGE PUMP

A charge pump circuit provides a voltage that is higher than the voltage of the power supply or a reverse polarity voltage. In many applications such as Power IC, continuous time filters, and EEPROM, voltages higher than the power supplies are frequently required. Increased voltage levels are obtained in a charge pump as a result of transferring charges to a capacitive load and do not involve amplifiers or transformers. For that reason a charge pump is a device of choice in semiconductor technology where normal range of operating voltages is limited. Charge pumps usually operate at high frequency level in order to increase their output power within a reasonable size of total capacitance used for charge transfer. This operating frequency may be adjusted by compensating for changes in the power requirements and saving the energy delivered to the charge pump.

Among many approaches to the charge pump design, switched-capacitor circuits such as *Dickson charge pump* [42] are very popular, because they can be implemented on the same chip together with other components of an integrated system. We use a 2-stage *Dickson charge pump* for generating higher voltage in our design (see figure 5.16.

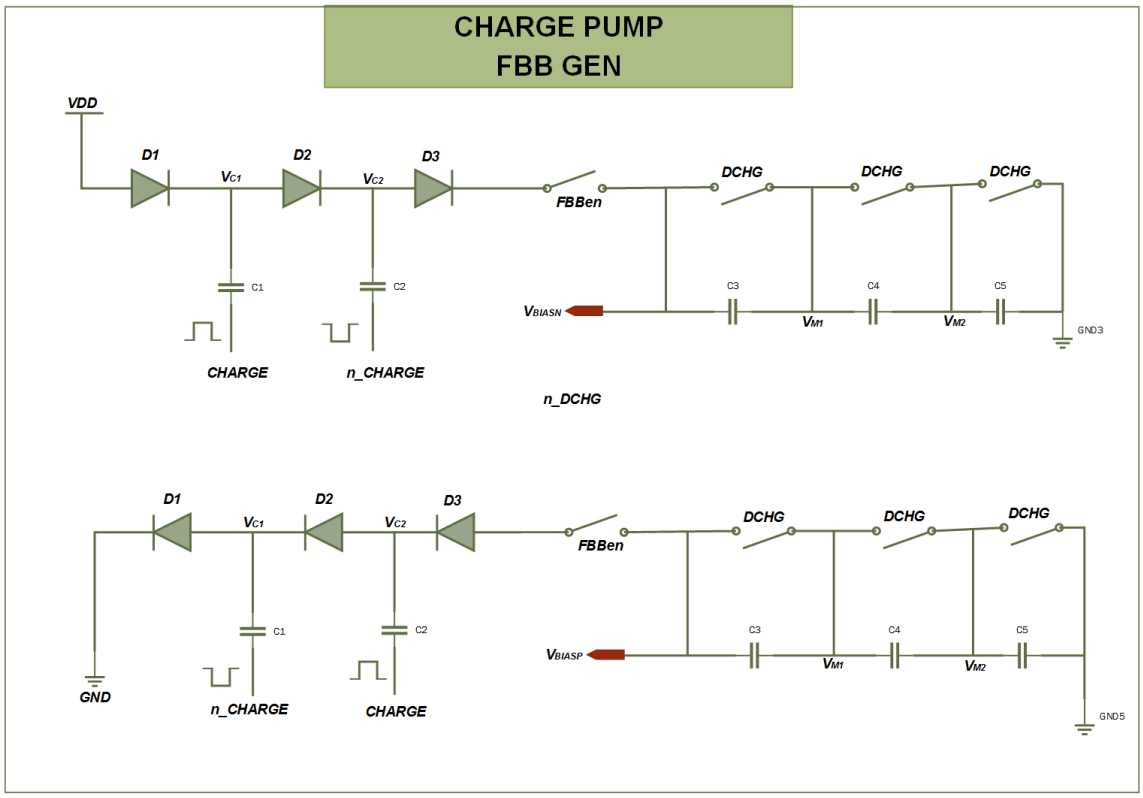

Figure 5.16: The proposed FBB & RBB Generator charge pump based on a Dickson Charge Pump 2- Stage

Figure 5.16 shows the proposed charge pump for PMOS transistors body bias and for NMOS transistors which is complementary to PMOS. The didoes are implemented by diode connected transistors. The circuit is a two-stage Dickson charge pump [42] and this kind of charge pumps are commonly used as voltage multipliers and negative voltage supplies. The circuit implemented is used to provide voltage lower then GND and higher than power supply voltage rails for FBB mode.

When *charge* signal is at low level C1 is charged, and hence V_{C1} goes to VDD -V_t. Afterwards, when *charge* signal goes to high level, V_{C1} doubles the high voltage level of the circuit and C2 is charged at 2(VDD -V_t). Finally, when *charge* signal goes back to low level V_{C2} rises up to 3(VDD -V_t). D1 and D2 prevent the discharge of C1 and C2 through VDD and V_{C1} respectively. As *charge* and *discharge* signal pulse widths depend on the phase error between the expected and measured delay on the VCDL. The width of the charge pulse increases as the delayed reference pulse becomes faster then the reference pulse and vice versa. When the VCDL achieves expected delay, the circuit stops pumping charge and leaves the charge pump output (V_{BIASP} , V_{BIASN}) in steady state.

The choice of capacitor values is a trade-off between area overhead, response time and body bias voltage accuracy: the bigger the capacitors, the higher the accuracy and area overhead. During reset or sleep periods the NBB signal is high and FBB signl is low. The switch of the floating transmission gate (see Figure 5.17) is open and the charge pump no longer charges the output capacitors C3,C4 and C5. After the system comes out of sleep

mode the FBB signal is high and the charge pump start charging the output capacitors. The wider the charge pulse the faster it charges the capacitors. Due to the diode losses the maximum output of the circuit is slightly greater then VDD. These diodes are implemented by means of diode-connected NMOS transistors.

When the main test circuit is in FBB mode and it is working with no errors, then after a certain defined monitoring period when the counter starts to decrement and reaches a count value of '0'. The reference pulse generated for count value '0' makes the phase detector to change from FBB to NBB mode,(switch FBB_{en} is opened) and thus C3,C4 and C5 capacitors are no longer charged. Two floating transmission gates slowly discharge these capacitors according to the phase shift between the delay line and the reference pulse width; the slower the VCDL delay output, the wider the DCHG pulse, and consequently faster the body bias decrease from maximum FBB to NBB mode. At least two transmission gates in series are needed in order not to exceed Drain-Source Breakdown Voltage and to make the discharge faster. Each of them discharges capacitors C3,C4 and C5 respectively.

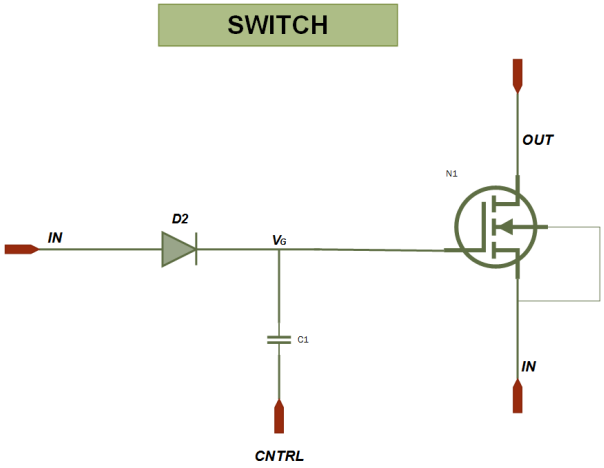

Figure 5.17: Switch: Floating transmission gate

As shown in Figure 5.17, these floating transmission gates are switched by means of a capacitive coupling since transistor gate is no longer referenced to ground (transistor source is at higher level). When gate terminal is set to high, the coupling capacitor produces a V_G voltage which VDD -V_t volts higher than the internal transistor source, and therefore, drives this transistor from cut-off to saturation. The main difference between floating transmission gates in Figure 5.16 is the W/L ratio of the internal transistor; while first switch is expected to act as a switch (fast transistor, large ratio), the remaining two switch are expected to produce a small decrease in body bias voltage (slow transistor, small ratio).

5.11. TEST PATTERN GENERATION

In order to test operation of the main test circuit the input vectors are generated by using linear feedback shift resistors(LFSR). The most commonly used linear function of single bits is exclusive-OR (XOR). Thus it is implemented using the shift resister whose input bit is driven by the XOR of some bits of the overall shift register value. As per the main test circuit the input *a* and *b* vectors generated are 4-bit.

6

SIMULATION SETUP AND RESULTS

6.1. INTRODUCTION

First we look into the types of 28nm- UTTB FDSOI devices used for the entire simulation of the proposed body bias generator.There are two standard V_{th} devices: one is 1.0 V low V_{th} transistors (LVT) seated on flip-Wells enable to apply high forward back-biasing (FBB) up to 3V. Another is 1.0 V regular Vth transistors (RVT) built on classical wells; enable strong reverse back-biasing (RBB) down to -3V. Simulation and design of 28-nm UTBB-FDSOI technology is done by using the High-k dielectric and metal gate electrode 1.0 V LVT from ST-microelectronics. The advantage of using LVT is low V_{th} compared to RVT (V_{th} = 0.34 V for LVT and V_{th} = 0.41 V for RVT). All the simulations are performed at 25^0C. The device used has a high-K dielectric value with ultra-thin silicon film of 7 nm. The ultra thin BOX has a thickness of 25 nm [4].

6.2. BODY BIAS (BB) CONDITIONS:

The simulations are carried out at power supply voltage of VDD = 1 V and 0.9 V , GND = 0 V. The main Idea for the adaptive design of the circuit is to change the body bias voltage to alter the speed of the circuit and also power consumption. The 28-nm UTTB FDSOI device LVT devices are qualified up to 1.3 V of FBB (V_{bn} > gnd , $V_{bp} \leq$ gnd). When increasing or decreasing the body bias voltage we should make sure it is within the qualified range of the UTTB-FDSOI device. So considering the body bias voltage range, the following BB conditions are defined:

1. **No Body Bias (NBB) Mode (V_{bn} = 0 V , V_{bp} = 0 V)** In the no body bias mode the NMOS and PMOS transistor body to source voltage(V_{bp}) is equal to *gnd*.

2. **Maximum FBB Mode (V_{bn} = 1.2V , V_{bp} = -0.3V)** In the maximum FBB operation mode observed in the simulations shown in this chapter, the PMOS transistor body to source voltage(V_{bp}) is decreased below **GND** voltage and the NMOS transistor body to source voltage(V_{bn}) is increased above **VDD** voltage.

6.3. SIMULATION TOOL AND SETUP

First, we augment the main test circuit, to do this, we determine the critical paths by using *Cadence Virtuoso* simulations. The digital design of the thesis is done by writing *VHDL* code

and then using *RTL cadence compiler* for synthesis step. The synthesis step transforms the hardware description language into gate-level net-list, given all the specified constraint like timing and design constraints and optimization settings (see the VHDL code in Appendix section A). The generated net-list is imported in the Cadence Virtuoso tool to perform the simulation of logic circuits. The flip-flops at the output of the critical paths are replaced by canary circuits (consisting of a main FF and a canary FF), whose structure and operation are described in section 6.3. Once the placement of canary circuits is done, then all the error signal are **ORED** together. Then the *timer unit* monitoring period is fixed to **250 ns** duration. This timer is configurable and can be changed depending on the design requirements. After this the control unit with 2-bit *UP-DOWN counter* is setup which records the number of errors from the test circuit. This makes the *reference generator block* to output reference pulse depending on the counter value. After this the important task is to fix the number of VCDL which is performed as described in section 5.8.

When the circuit is powered on initially it will be in reset condition i.e in sleep mode for **100 ns**. During this period the charge pump, capacitor is charged up and output ramps from initial level to final regulation level. After a delay of 100 ns, circuit comes out of *sleep mode* and enters the *operation mode*.

6.4. SIMULATION RESULTS

6.4.1. TIMING ANALYSIS

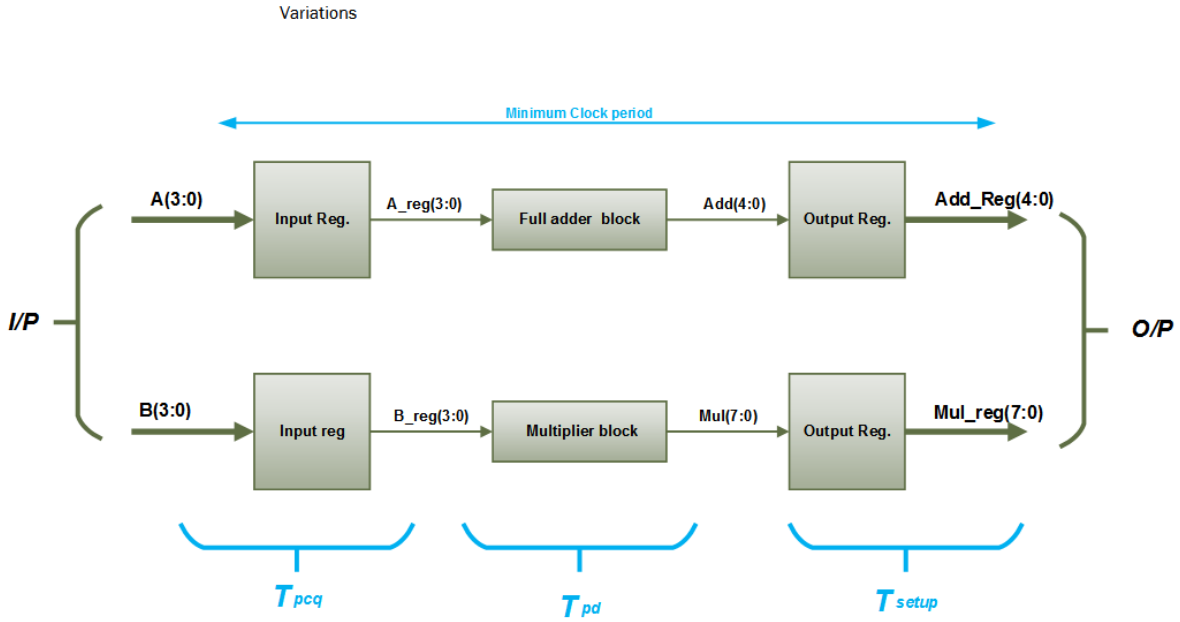

Figure 6.1: Min clock period- Proposed Main test circuit

The different types of timing analysis are validated for the main test circuit (see figure 5.2). Ideally for the combinational part the entire clock cycle will be available, if the combinational logic delay is too great, then the receiving element will miss its set-up time and sample the wrong value. This is called a *set-up time failure* or *max-delay failure*. It can be

solved by redesigning the logic to be faster or by increasing the clock period. This section computes the actual time available for the combinational part and D-Flip-Flops.

The figure 6.1 shows different timing parameters calculated. The adder block uses 4 full adders(FA), and array multiplier which is a 4 bit uses n $*$(n-2) FA and n HA. So for n = 4-bit, we get 8 FA and 4 HA. Hence the maximum delay(see fig) i.e time taken for the operation to complete is for the array multiplier. The delay associated with the array multiplier is the time taken by the signals to propagate through the AND gates and adders that form the multiplication array as shown in figure 5.3. Delay of an array multiplier depends only upon the depth of the array not on the partial product width. The maximum delay constraint i.e minimum clock period required for the correct operation of the entire circuit (see figure 6.1) can be divided into three parts:

1. **Calculation of D-flip flop Propagation time(T_{pcq}):** First we calculate the input clock-to-Q propagation time of the registers i.e D-flip flops as shown in figure 6.1.the D-flip-flop time required for the input data to propagate from input to output. As per the figure 6.2. The measured time T_{pcq} is **30 ps**.

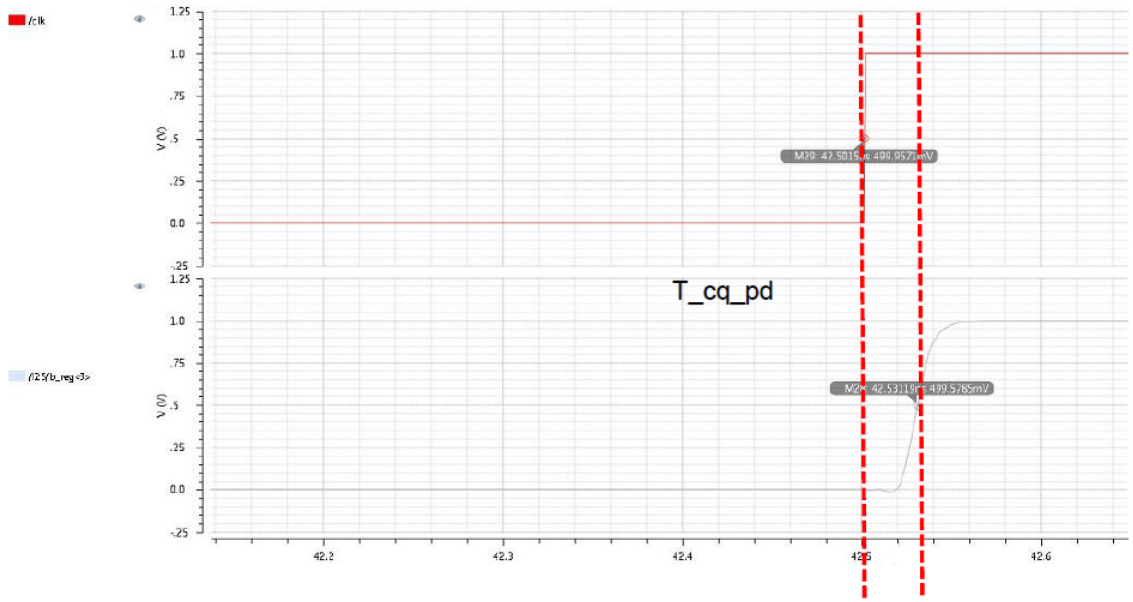

Figure 6.2: D-flip flop Propagation time(T_{pcq})

2. **Calculation of flip-flop set-up time(T_{setup}):**

   This is an important timing parameter to calculate the flip-flop setup time **T_{setup}** which is the minimum time required for the data to get stabilized before clock transition. If the output from the combinational part is within the minimum T_{setup} then the there is a valid output(figure 6.3). Otherwise if the combinational part takes more time such that the minimum T_{setup} is not met then the output part will get the error data as shown in figure 6.4. From the figure 6.3 shows the minimum T_{setup} required by D-flip flop which is **min 7 and max 10 ps**.

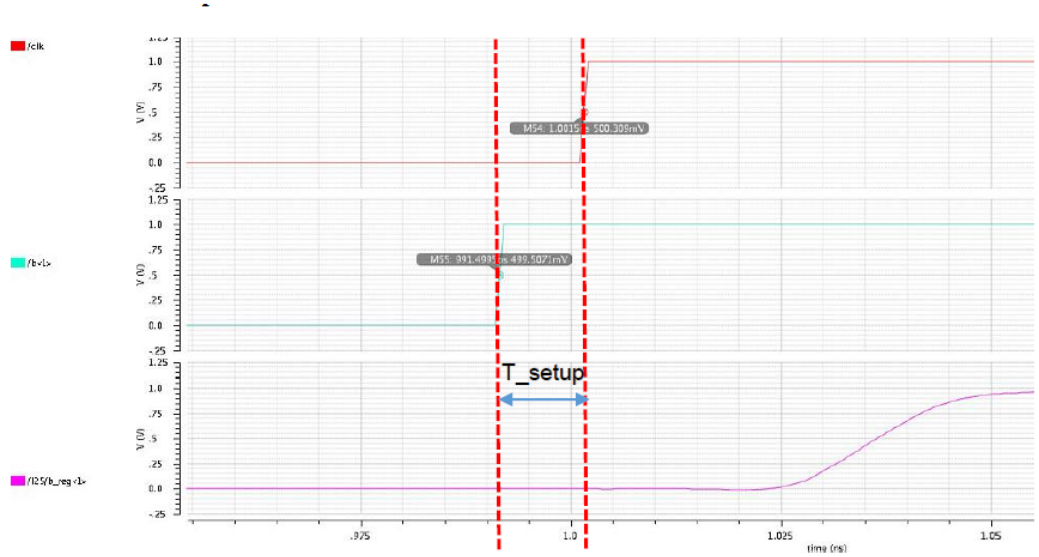

Figure 2: Min setup time for D-FF

Figure 6.3: D-flip flop Set-up time(T_{setup})

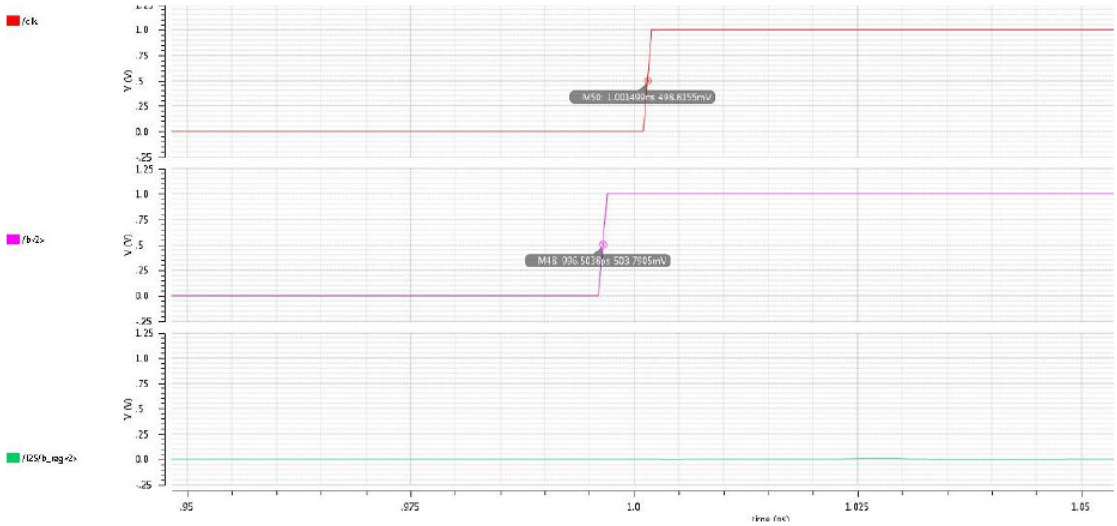

Figure 6.4: D-flip flop set-up violation

3. **Calculation of Combinational path maximum delay (T_{pd}):**

   Similarly for the combinational part, the multiplication path has the maximum propagation delay which is also the **critical path** delay. It is defined as the path with maximum delay from input to output port. So we consider the said part to calculate the delay. Figure 6.6 shows the time vs 8-bit multiplier nets. The net *mul_6* is the path with maximum delay (see fig 6.5 shows the traverse of the path) when compared to remaining nets. Hence the maximum combinational delay (t_{pd}) is **257.57 ps** (see Fig-

ure 6.6).

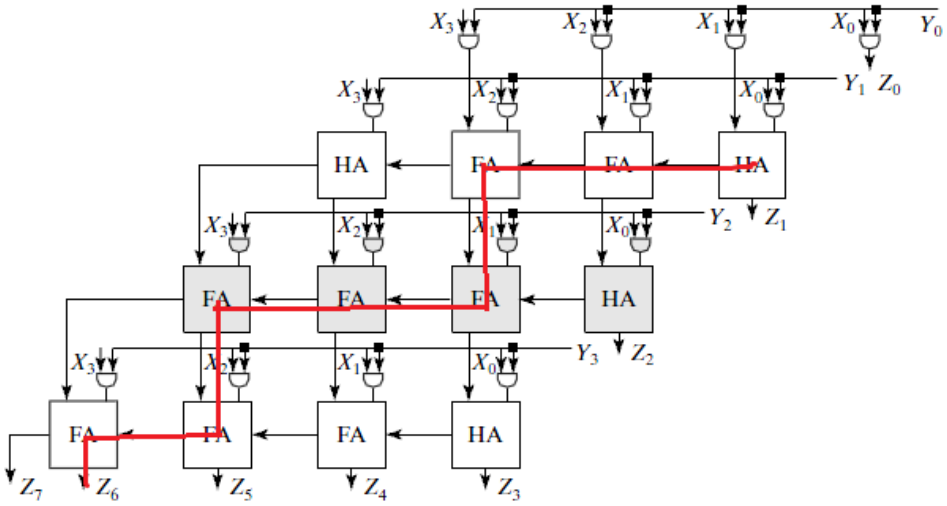

Figure 6.5: Conventional 4X4 array multiplier with maximum delay path

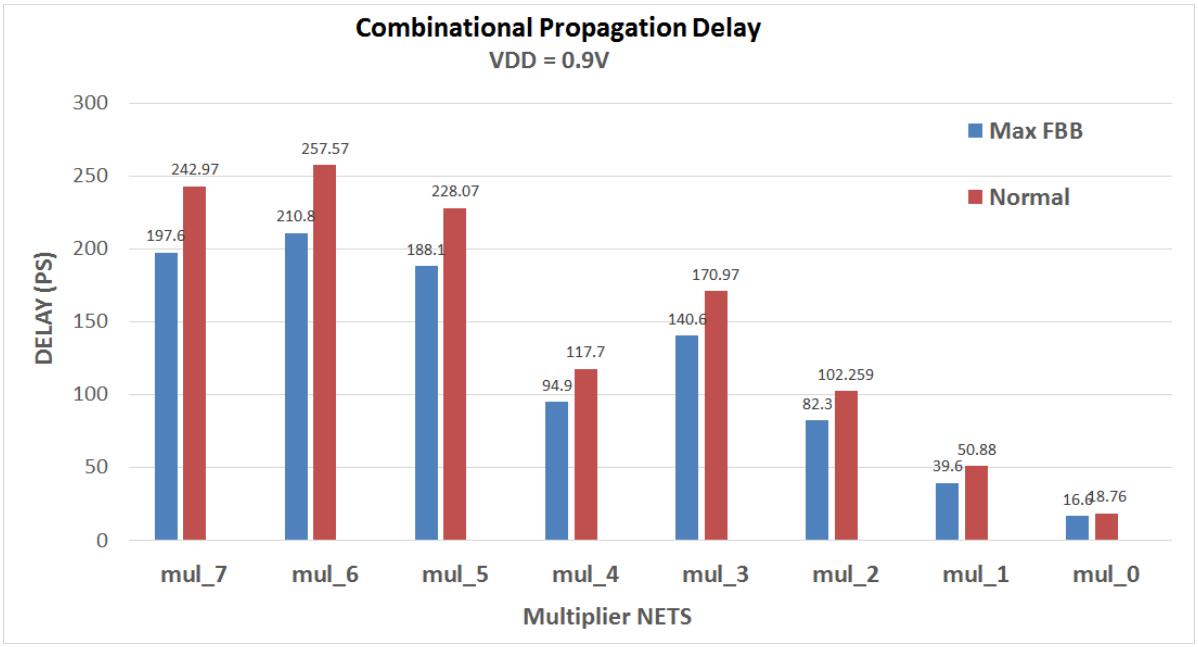

Figure 6.6: Combinational path maximum delay (T_{pd})

Figure 6.7, shows the **Critical Path** taking into consideration the three BB conditions as defined in experiment set-up section, and performing simulations (with VDD = 0.8V,0.9V,1V). As we see from the figure for VDD = 1V, *NBB* mode has the maximum critical path. Since in NBB mode the threshold voltage (V_{th}) is more then the FBB mode and thus gate delay, hence reducing the leakage current (saves power consumption mainly when the circuit is in reset or sleep mode).

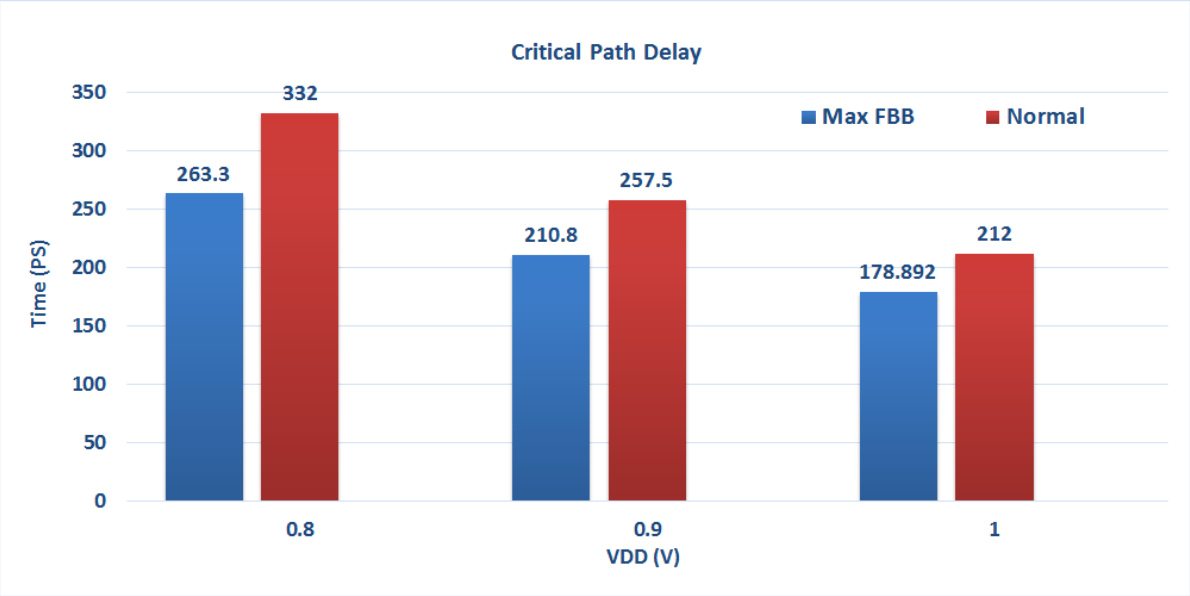

Figure 6.7: Critical Path: three BB conditions and VDD = 1V, 0.9V, 0.8V

6.4.2. MINIMUM CLOCK FREQUENCY CALCULATION

After measuring important timing parameters t_{pcq}, t_{pd}, t_{setup} which are sufficient requirement to calculate minimum clock period (clk_{min}). The timing equation to calculate the clk_{min} is defined as:

$$T_c = t_{pcq} + t_{pd} + t_{setup} \tag{6.1}$$

$$clk_{min} = 30ps + 212ps + 10ps = \mathbf{252\ ps}$$

This minimum clock period required is validated by running the simulations with a predefined VDD, to the main test circuit and not using any other sections from the design (without Canary logic, control unit, BBG, VCDL and reference generator). For this VDD, we run the simulations to find out the nominal clock period such that no error occurs during testing phase. We add a safety margin of 15% to this period and the resulting value becomes clock period for our simulations. For a VDD of 1V, this final value is found to be 300 ps (f

= 3.33 GHz) and 375 ps (f = 2.67 GHz) for a VDD of 0.9V respectively (see figure 6.8). All simulation are done in normal BB operation mode (V_{bn} = 0 V , V_{bp} = 0 V).

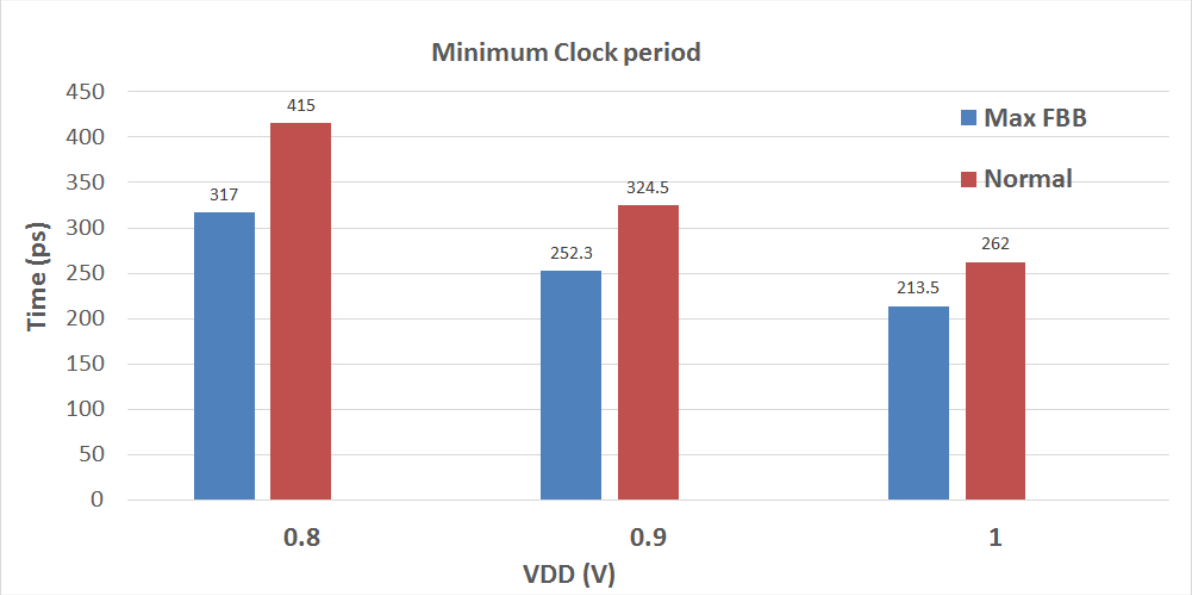

Figure 6.8: Clock period: three BB conditions and VDD = 1V,0.9V,0.8V

Since the proposed adaptive body bias unit does not need to operate with high safety margins, the clock period is set to be 300 ps and for this clock period minimum VDD is determined such that no error occurs during testing the main test circuit without other blocks. For this benchmarks, VDD is set to 0.9V for the entire system. As seen in figure 6.8 a histogram plot, when the system is operated at VDD = 0.9V, with clock frequency of 300 ps (as for normal BB operation the clock frequency required is 324.5 ps see figure 6.8 histogram plot), the circuit will operate in FBB condition. But the data path performing operations not always uses the critical path. Hence we can say that the time duration for the circuit to be in FBB mode is small as the operation is carried out. In FBB mode(see figure 6.8) the frequency required is 252 ps which satisfies the calculated clock frequency of 300 ps. From here onwards we will use the clock frequency at 300 ps and VDD at 0.9V for all other simulations carried on the adaptive BB system.

6.4.3. CANARY LOGIC SIMULATION WAVEFORM

Canary FF technique implementation in section 5.4 tells about the advantages. But the occurrence of timing errors cannot be completely eliminated because canary FF can only *predict* the occurrence of timing errors and the prediction is not always guaranteed. Therefore, to apply canary FF technique to practical applications, the occurrence rate of timing errors must be quantitatively assured.

In the first waveform the error is predicted correctly before it going to happen (figure 6.9 see the *canary_out* and *final_output* data signal). When the two paths have a mismatch in data which occurs i.e. *canary_out* has data 153(decimal) instead of 121(decimal), at this point **Error** signal is logic high, means the error is predicted correctly before it happens. This allows adaptive system to change the BB voltage such that the circuit enters the FBB mode. In the next clock cycles the circuit becomes faster so that the error is avoided.

But always the error are not captured correctly as shown in figure 6.10. See figure 6.10 when *input_a* is 13 and *input_b* is 14 ,the final product after multiplication is 182(decimal). But the final output signal at the rising edge of the clock has captured the product as 214(decimal) instead of 182(decimal). Hence this kind of output data errors can't be predicted by canary logic (this is one of the disadvantage, here in these situation razor circuit will be effective [3]).

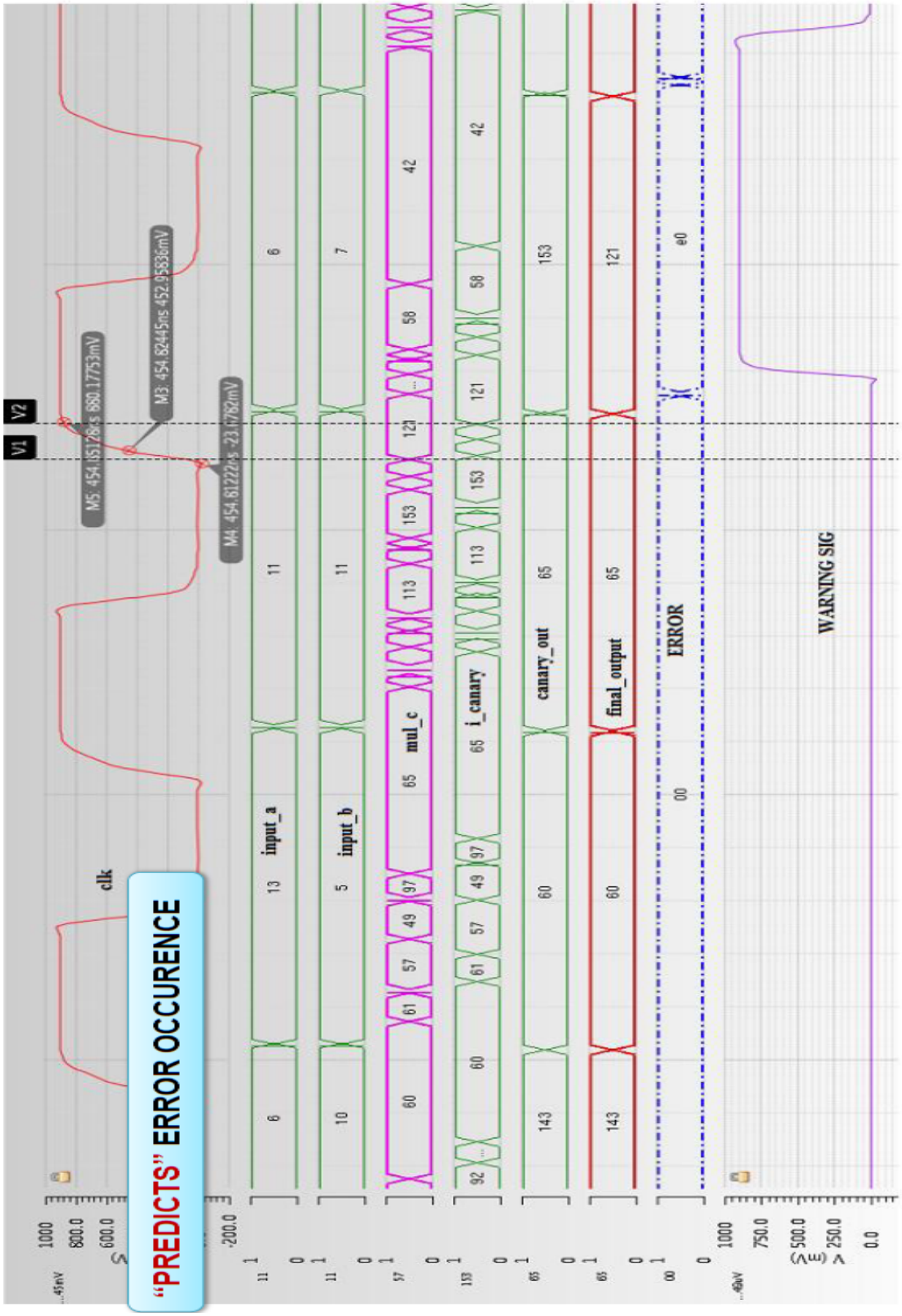

Figure 6.9: Canary Logic: Captured Error prediction waveform

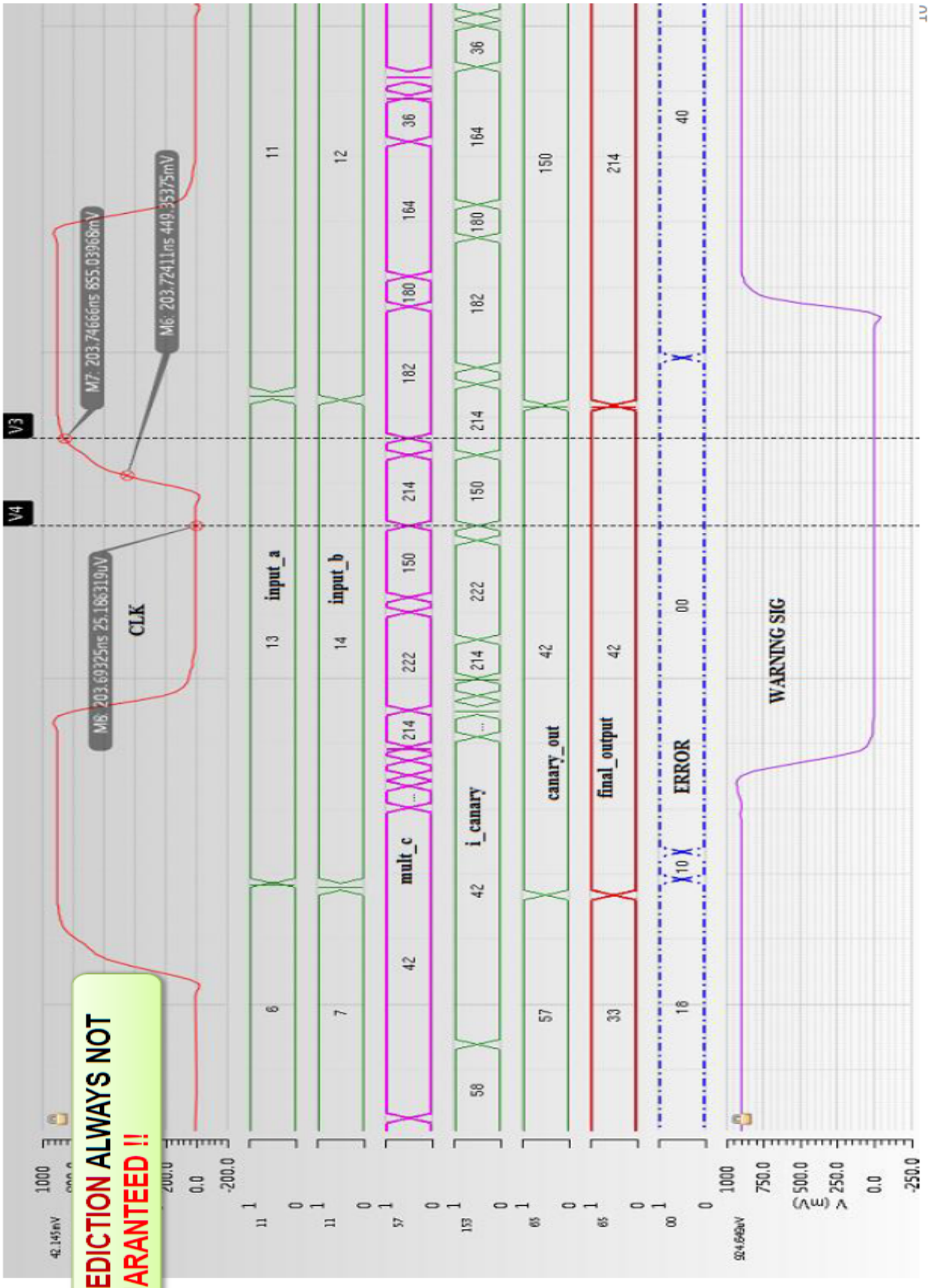

Figure 6.10: Canary Logic: Missing error capture

6.4.4. REFERENCE PULSE GENERATOR SIMULATION

In the reference pulse generator, the duty cycle of the pulse width depends on the error count from the control unit. The counter values decides the circuit bias conditions. In the design proposal we have used a 2-bit UP-DOWN counter, hence we can go for categorizing the four count values from level 0 to 3 as explained below:

- Level 0 is the No body bias (NBB) condition; levels 1 to 3 are in the increasing order of the forward body biases(FBB).
- Since intrinsic process variations are more or less, that remain constant throughout the lifetime of the chip, a single body bias level is sufficient to correct them. However, when dealing with voltage ,temperature variations, which monotonically degrades the circuit performance with time, forward body bias is necessary to restore the circuit performance.
- To be able to handle both cases efficiently, an up counter is used that counts upward (increases forward bias) when a warning signal is generated by the canary circuit. It counts upward till it reaches the highest forward body bias state (binary 11 in our case) and freezes in that state.
- Similarly for the NBB condition, the counter is decremented when there is no warning signal generated. If there are no warning signal generated within a certain duration of time, the monitoring unit asserts a timeout signal after a period of 250 ps. Then the counter is decremented from level 3 (binary 11) to level 0 (binary 00) for each timeout signal of 250 ns. Thus the decrement counter makes the circuit enter the NBB mode to save power.
- A four-state counter is implemented as a few forward body bias levels and reverse body levels are sufficient for the proposed circuit.

The figure 6.11 shows the DPWM reference pulse for 4 counter values, as each counter generates different pulse width. The pulse width decreases as the counter value increments from '0' to '3'. This decreasing pulse width is implemented to make sure the circuit operates in FBB mode (see section 5.9 for Phase detector operation)

Table 6.1: Body bias modes:(**NBB** = no body bias, **FBB** =Forward body bias, **VDD** =V_{supply})

Level	**Bias Mode**	**Condition**
0	NBB	V_{bn} = gnd, V_{bp} = gnd
1	FBB (Min.FBB)	V_{bn} > gnd, V_{bp} < gnd
2	FBB (Medium FBB)	V_{bn} > gnd, V_{bp} < gnd
3	FBB (Max.FBB)	V_{bn} = 1.2 V, V_{bp} = -300 mV

BBG SIMULATION WAVEFORM

The BBG block generates the body bias voltage required for the main test circuit. As in figures 6.12. and 6.13. shows the transient simulation waveform for 0.5μs duration. At the beginning of the simulation the circuit is forced to enter reset or sleep state for 100 ns and

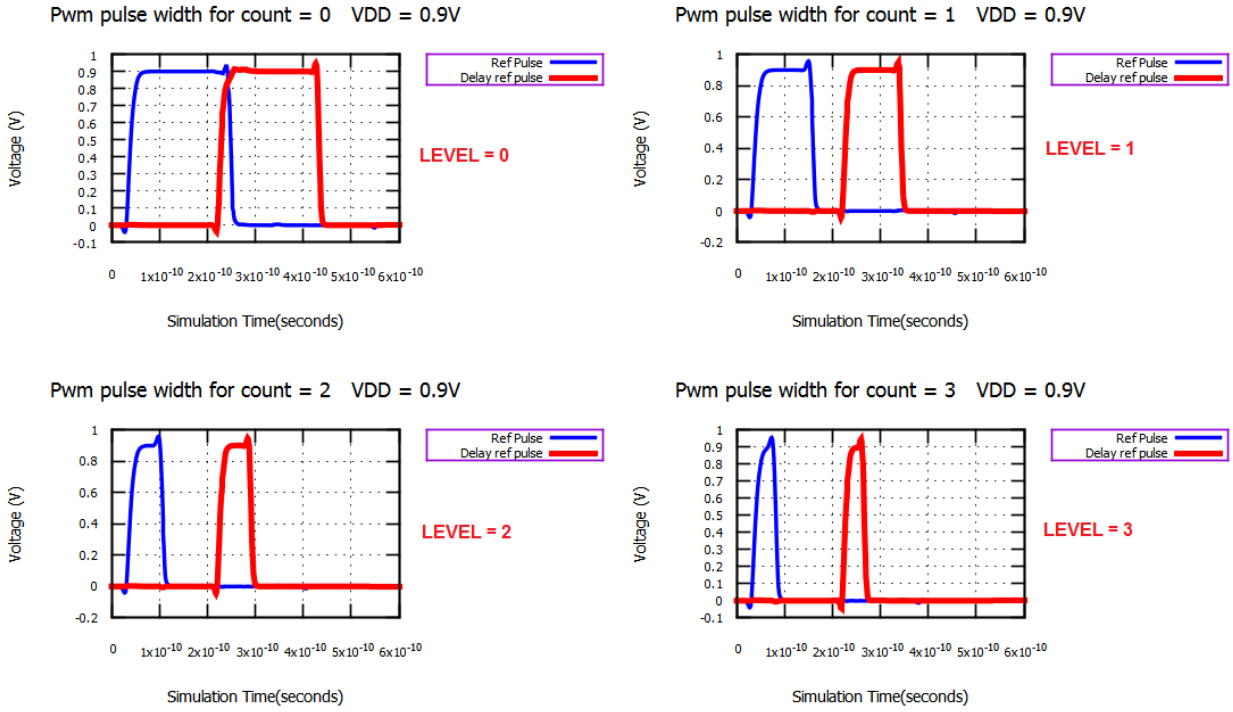

Figure 6.11: DPWM waveform for counter values = 0 to 3

NBB mode is enabled. During sleep state the NBB switch is enabled which keeps the bias voltage to V_{bn} = gnd, V_{bp} = gnd (see figure 5.12). In this state the Dickson charge pump is not charging but the output capacitors C3,C4 and C5 (see figure 5.16) are discharged. As a consequence, the main circuit including the VCDL transistors increase their V_{th} compared to FBB mode. This increase causes a reduction in leakage power consumption. It also produces a dramatic increase in the end-to-end delay, which is not relevant during sleep states.

Once the sleep state stops, the circuit tracks the input pattern pulse by charging the capacitors C1 and C2 (see Figure 5.16) and thus reducing bbiasP below *gnd* and bbiasN above *gnd* voltages. The normalized end-end to delay in the NBB mode has a maximum delay (see figure 6.13.), but as the circuit moves from sleep mode steady state which is the operating conditions after certain duration the normalized value is about 0.85, which means that the system is operating in FBB mode. As the VCDL acts like a delay sensor of the main circuit to generate the BB signals using a target delay as timing reference pulse, hence this delay is controlled by applying FBB voltage which

In conclusion if the normalized delay is equal to 1 then the circuit is in NBB mode and when the normalized delay is below 1 it enters the FBB mode. The complete simulation of the proposed system is shown in figure 6.14 6.15. The waveform shows the test circuit,canary,counter data signals and the remaining signals are from the BBG block.

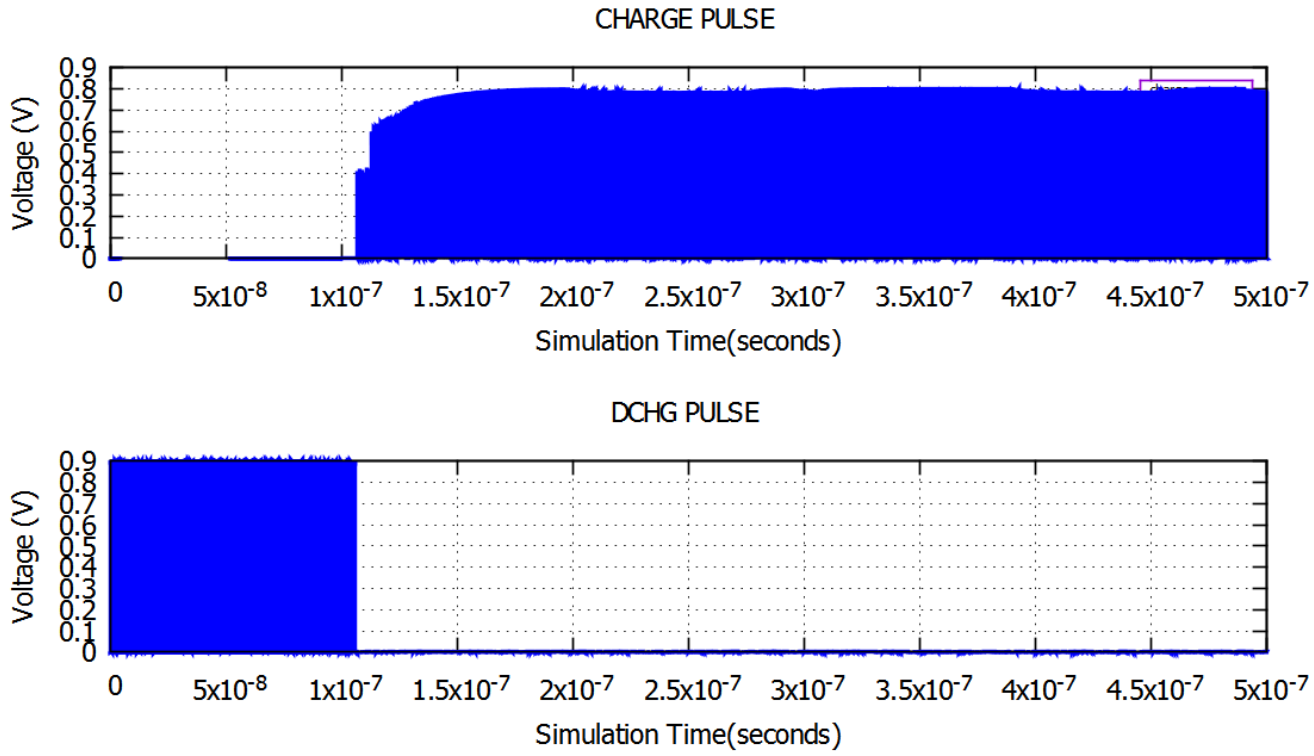

Figure 6.12: Charge pulse and discharge pulse simulation waveform

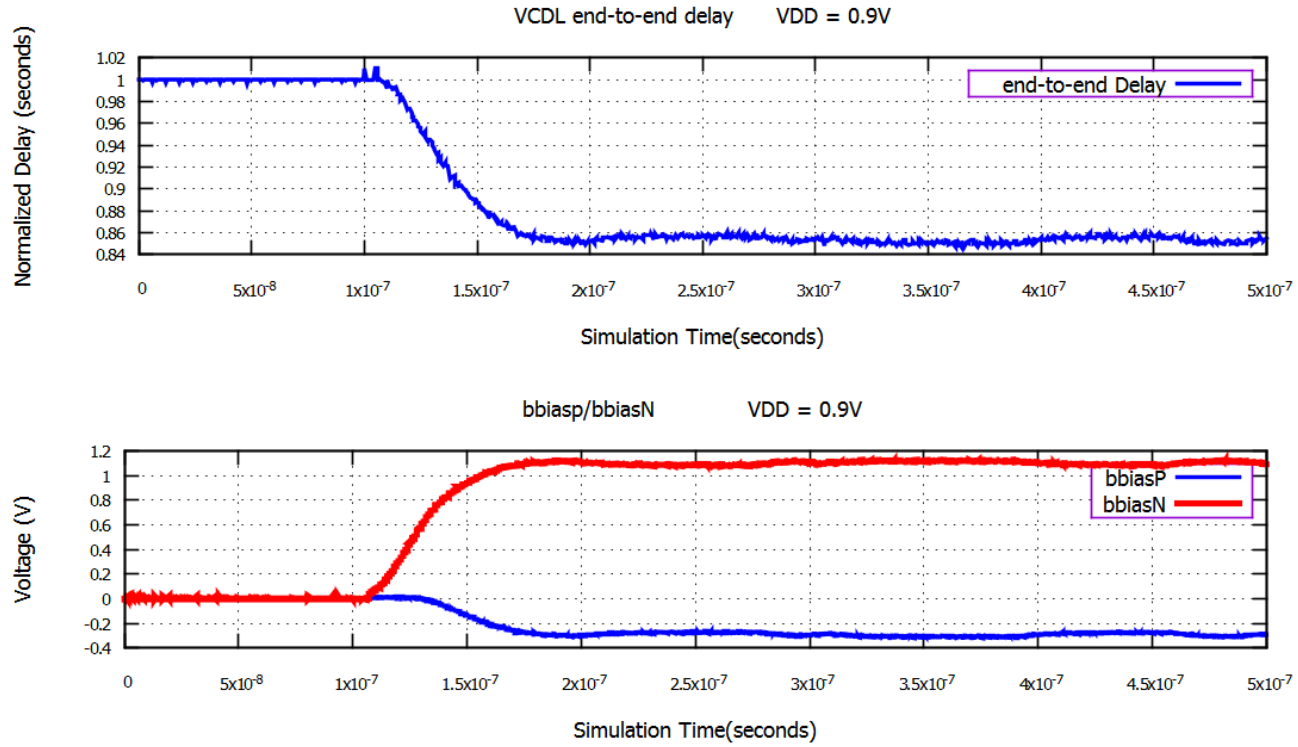

Figure 6.13: VCDL Normalized : End-end delay simulation waveform

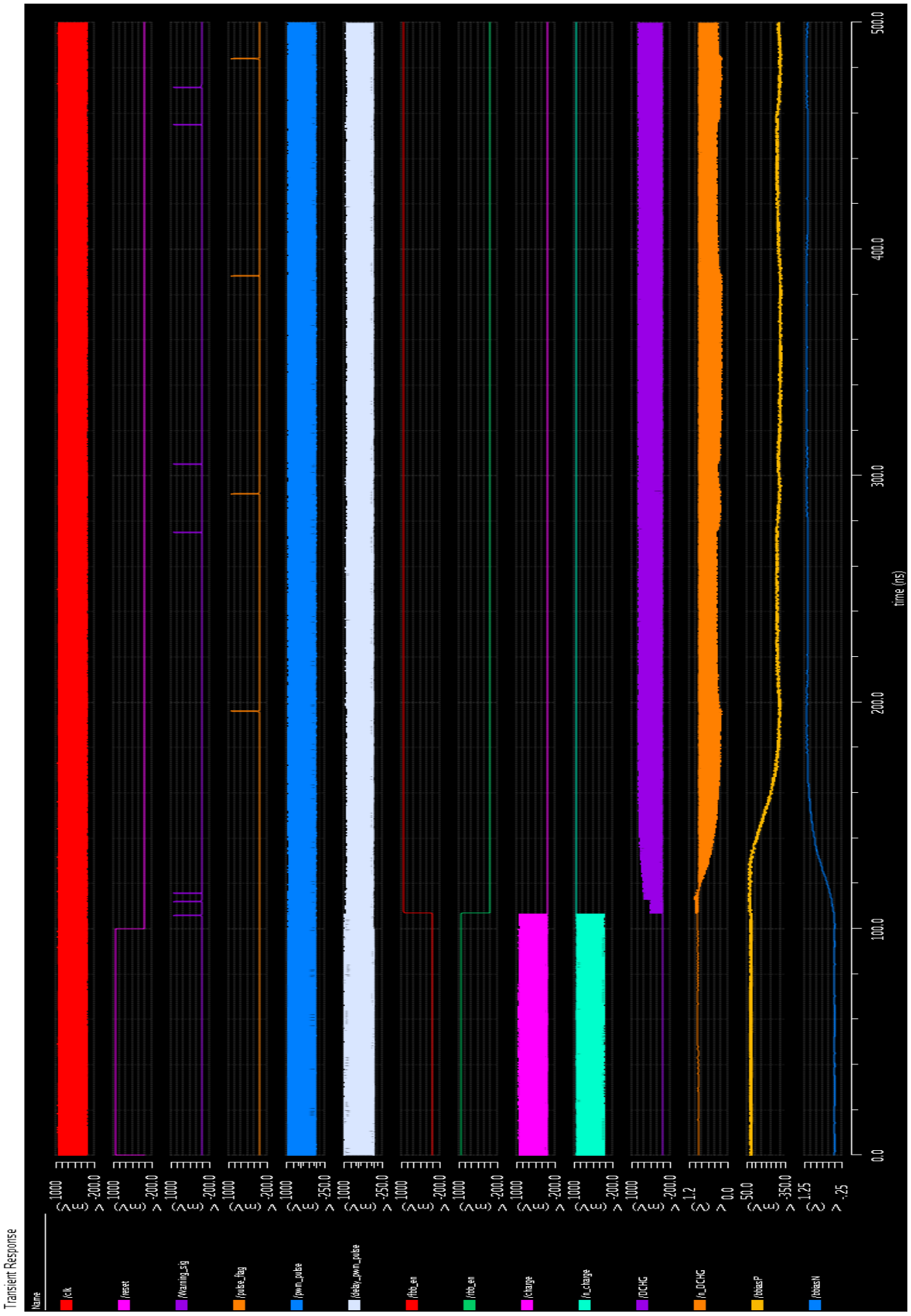

Figure 6.14: Complete adaptive body bias system simulation waveform 1

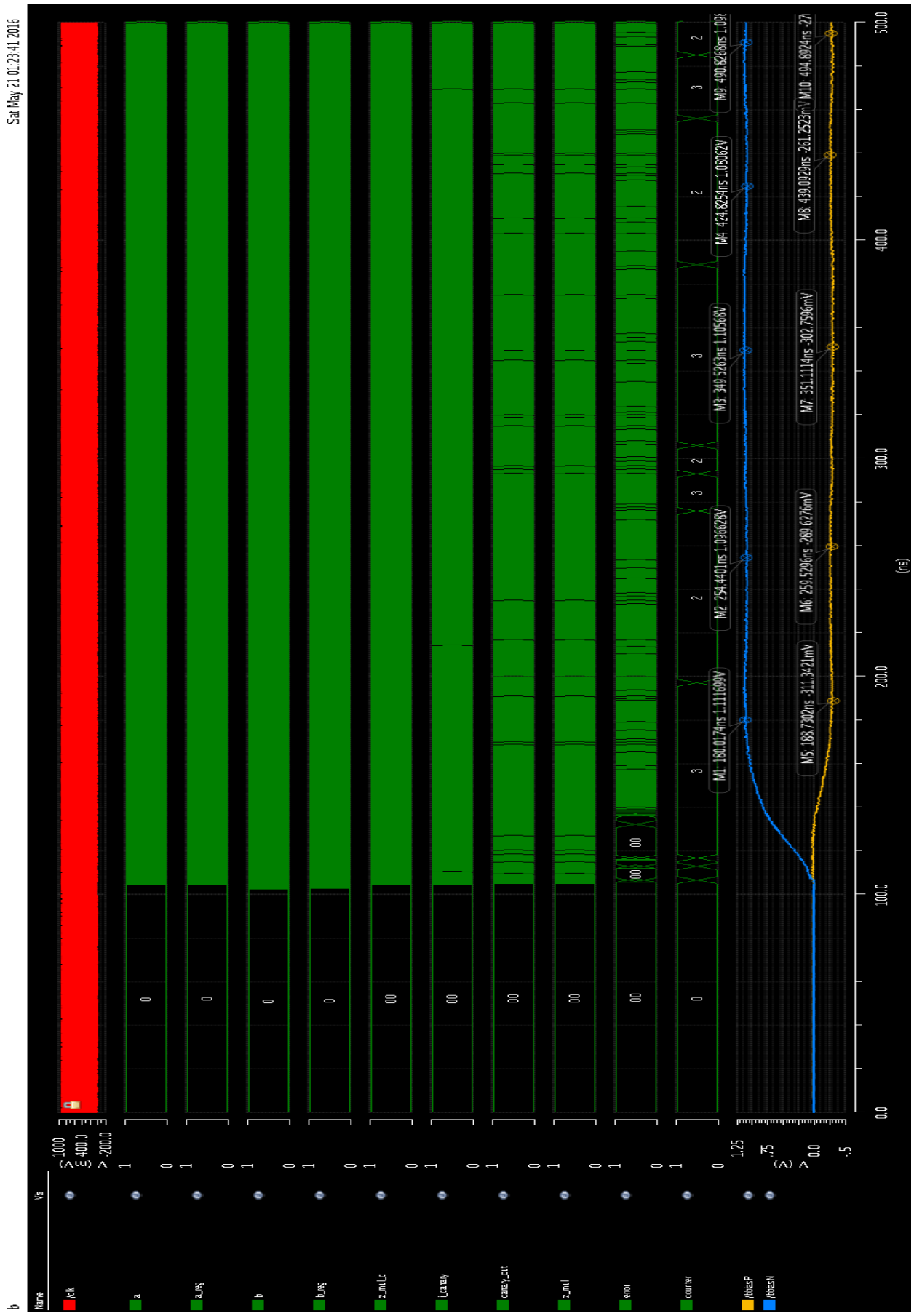

Figure 6.15: Complete adaptive body bias system simulation waveform 2

6.5. TOTAL LOGIC GATE USAGE

As explained in the section 6.3 about the cadence RTL synthesis which involves converting the RTL to generic gates and registers. The table 6.2 list out the number of logic gates required to implement the different blocks of the system. The digital logic is constructed with basic combinational gates like INV, AND, OR, NAND, NOR , sequential cells like Flip-Flops, latches. From the table 6.2 and figure 6.16 the total number of logic gate used for the complete system is **262**. The main test circuit which is a basic 4-bit adder and multiplier uses *54* logic gates. The Canary logic block, Control unit, reference pulse generator block and body bias (BB) generation block uses *208* gates. Therefore **20.6**% of gates are used by the main test circuit and remaining **79.3**% is used by the remaining blocks.

As said in the design approach section 5.1 we are using a simple test circuit to demonstrate the adaptive BB system. For a good comparison in power consumption while using the proposed system we have to consider a complicated circuit which uses hundreds of logic cells for example,a FPGA spartan LX45 part uses 27,000 logic cells. Even if we implement the test circuit using 16-bit and 32-bit array multiplier, which uses 169 and 594 logic cells respectively.

Table 6.2: Logic Gates used in different block section

Blocks	**Number of Combinational Circuits**	**Number of sequential circuits**	**Total Circuits**
Main test circuit	32	22	**54**
Timer Unit	28	10	**38**
Control Unit	26	6	**32**
Canary Logic block	11	8	**19**
Ref. block	33	0	**33**
VCDL	24	0	**24**
LFSR	4	8	**12**
Delay Buffers	50	0	**50**
		TOTAL	**262**

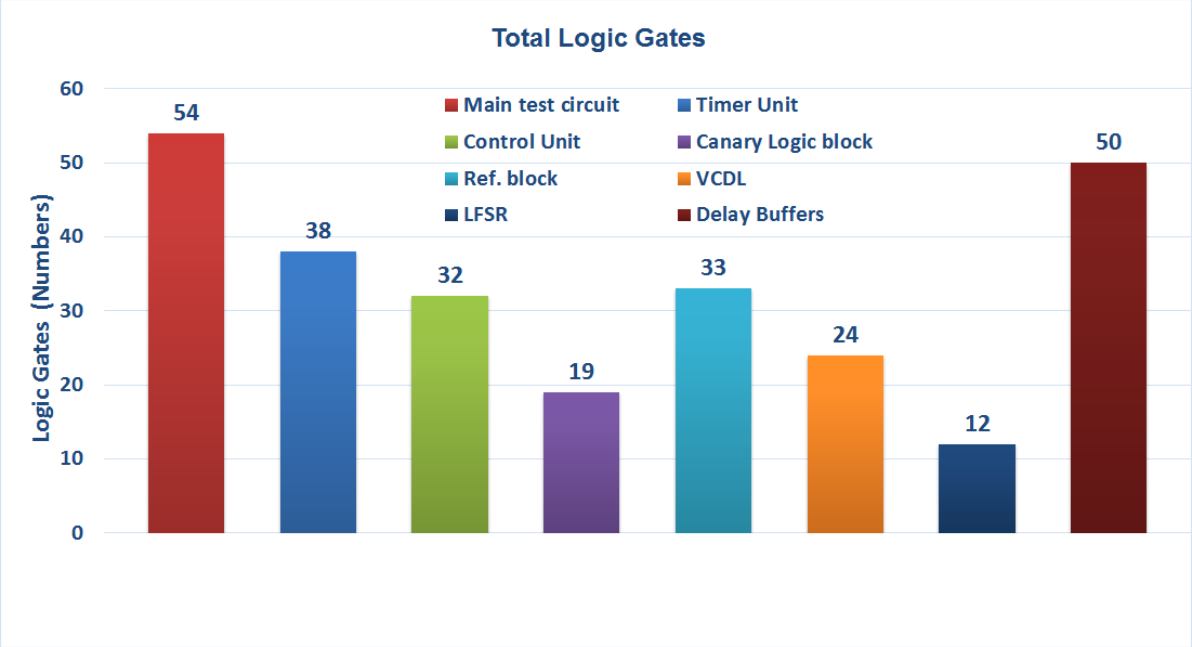

Figure 6.16: Total Logic Gates for different section of the system

7

CONCLUSION

In this thesis proposal, a novel reliable design technique of variability compensation architecture with adaptive body bias (BB) is presented. Adaptive BB system allows VLSI circuits to autonomously compensate for PVT variations. Being a typical-case design methodology and the ability to tune itself, it helps the designer to avoid the unnecessary safety margins in the design stage. This proposal exploits the advantage of excellent body bias range and its advantages as compared to the bulk CMOS, hence a better BB control. The following can be concluded from the thesis proposal:

- The design is able to use charge pump for body bias generation which replaces the conventional DAC but it also manages to generate body bias voltage greater then supply voltage rails.
- The proposed design also generates separate body bias control signals for NMOS and PMOS. This means that a careful design is needed to find the optimum relation between NMOS and PMOS body bias for maximum performance and minimum power.
- The circuit speed is controlled digitally as we are using a control unit to speed up and a timer unit to slow down.
- The proposed design uses a 2-bit counter which provides Coarse Grain Voltages for body bias. But this can be easily overcome by using a higher bit counter and go for fine grain body bias regulation.
- The addition of error detection and adaptive body bias introduce an overhead in area and power, especially for small circuits like the test bench here presented. Due to the long simulation times (several hours) the power could not be reliably determined and is not included in the report. A more careful evaluation of power is needed to fully determine the advantages of the circuit.
- The area overhead for the canary delay buffers can be overcome by using configurable Canary circuits such that the inserted location and the buffer delay can be configured. Other option may be to implement the Canary circuits only to Critical paths and higher nibble bits. These two options reduce both power and area consumption.

- A significant portion of the power consumption overhead is due to the body bias generator. This overhead can be reduced by implementing a gating option which enables the FBB voltage for a certain duration and then disables the control signals which consume more switching power. By doing this we can reduce the power.

A

APPENDIX- VHDL CODE

A.1. UP-DOWN COUNTER

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_std.all ;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity up_down_counter is
generic (N : natural := 2);
  port (
    Incr        :in  std_logic;                    -- up_down control  counter
        Decr    :in  std_logic;
    clk     :in  std_logic;                    -- Input clock
    reset   :in  std_logic ;                   -- Input reset
        cout    :out std_logic_vector (N-1 downto 0)   -- Output of the counter
  );
end entity;

architecture rtl of up_down_counter is

signal count :std_logic_vector (N-1 downto 0);

begin
process (clk, reset) begin

    if (reset = '1') then
        count <= (others=>'0');
    elsif (rising_edge(clk)) then

                if (Decr = '1' and count = x"0" ) then--let the counter be in RBB
                    mode when there are no errors from the system
                        count <= (others=>'0');

        elsif (Incr = '1' and count < "11") then
            count <= count + 1;
        elsif (Decr = '1') then
            count <= count - 1;
                elsif(Incr = '1' and count = "11") then
                        count <= "11";
```

```
                end if;

    end if;
 end process;
    cout <= count;
end architecture;
```

A.2. MAIN TEST CIRCUIT UNIT

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_std.all ;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

--ENTITY DECLARATION: name, inputs, outputs
entity andGate is
   port( A, B : in std_logic;
         F : out std_logic);
end andGate;

--FUNCTIONAL DESCRIPTION: how the AND Gate works
architecture func_andgate of andGate is
begin
  F <= A and B;
end func_andgate;

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_std.all ;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

Entity fulladd is
        port ( a, b          : in std_logic;
                    Cin                  : in std_logic;
                    sum,Cout       : out std_logic
        );
end fulladd;

Architecture fulladd_logic of fulladd is
begin
        sum <= a xor b xor Cin;
        Cout <= (a and b) or (Cin and a) or (Cin and b);
end fulladd_logic;

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_std.all ;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

Entity dff is
        port ( clk, reset      : in std_logic;
                    d                    : in std_logic;
                    q                    : out std_logic
        );
end dff;
```

```
Architecture dff_rtl of dff is
begin
process (clk,reset)
begin
        if(reset ='0') then
                q <= '0';
        elsif (clk'event and clk ='1') then
                q <= d;
        end if;
end process;
end architecture dff_rtl;




--logic impelmentation--------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_std.all ;
use IEEE.STD_LOGIC_UNSIGNED.ALL;


Entity simple_design_hier is
        port (  clk ,rstn,cin   : in std_logic;
                        a, b                    : in std_logic_vector (3 downto 0);
                        Cout                    : out std_logic;
                        z_add                   : out std_logic_vector(3 downto 0);
                        z_mul                   : out std_logic_vector(7 downto 0)

        );
end simple_design_hier;


Architecture simple_design_hier_structure of simple_design_hier is

        signal c                        : std_logic_vector (3 downto 1) := "000";
    signal a_reg                : std_logic_vector (3 downto 0);
    signal b_reg                : std_logic_vector (3 downto 0);
        signal z_add_c              : std_logic_vector (3 downto 0);
        signal z_mul_c              : std_logic_vector (7 downto 0);
        signal AND_out                          : std_logic_vector ( 35 downto 0) ;
        signal W                        : std_logic_vector (3 downto 0) ;
        signal Cin_reg,cout_reg         : std_logic;
        type array_1 is array (6 downto 0,6 downto 0) of std_logic;
        signal z_cin_zero               : array_1;
component fulladd
        port (  a, b            : in std_logic;
                        Cin                     : in std_logic;
                        sum,Cout        : out std_logic
                );
end component;

component andGate is
   port( A, B : in std_logic;
             F : out std_logic );
```

```
end component;

component dff is
port (  clk, reset      : in std_logic;
                        d                       : in std_logic;
                        q                       : out std_logic);
end component;

BEGIN


        W               <= ( others => '0' );

-----///D-flip flop assignemnts//----------------------

        d9 :  dff port map ( d => cin ,  clk => clk , reset =>rstn , q => cin_reg );
        d10 : dff port map ( d => cout_reg ,  clk => clk , reset =>rstn , q => cout
            );

        Gen1 : for i in 0 to 3 generate
        SM1 : dff port map ( d => z_add_c(i) , clk => clk , reset =>rstn , q =>
            z_add(i));
        end generate Gen1;

        Gen2 : for i in 0 to 7 generate
        SM2 : dff port map ( d => z_mul_c(i) , clk => clk , reset =>rstn , q =>
            z_mul(i));
        end generate Gen2;

        Gen3 : for i in 0 to 3 generate
        SM3 : dff port map ( d => a(i) , clk => clk , reset =>rstn , q => a_reg(i));
        end generate Gen3;

        Gen4 : for i in 0 to 3 generate
        SM4 : dff port map ( d => b(i) , clk => clk , reset =>rstn , q => b_reg(i));
        end generate Gen4;

----------ADDER IMPLEMENTATION---

        stage_0: fulladd port map (Cin => Cin_reg,a => a_reg(0), b => b_reg(0),  sum
             => z_add_c(0), Cout => c(1));
        stage_1: fulladd port map (Cin => c(1),   a => a_reg(1), b => b_reg(1),  sum
             => z_add_c(1), Cout => c(2));
        stage_2: fulladd port map (Cin => c(2),   a => a_reg(2), b => b_reg(2),  sum
             => z_add_c(2), Cout => c(3));
        stage_3: fulladd port map (Cin => c(3),   a => a_reg(3), b => b_reg(3),  sum
             => z_add_c(3), Cout => cout_reg);


        stage_5555: fulladd port map (Cin => z_cin_zero(6,1),   a => a_reg(3), b =>
            b_reg(3),  sum => z_add_c(3), Cout => cout_reg);

----------MULTIPLIER 4x4 IMPLEMENT---
        AND1 : for i in 0 to 3 generate
        AND_1 : andGate port map ( A => a_reg(i),  B => b_reg(0),F => AND_out (i));
        end generate AND1;
```

```
        AND2 : for i in 0 to 3 generate
        AND_2 : andGate port map ( A => a_reg(i), B => b_reg(1),F => AND_out (i + 4
             ));
        end generate AND2;

        AND3 : for i in 0 to 3 generate
        AND_3 : andGate port map (  A => a_reg(i), B => b_reg(2),F => AND_out (i +
            8));
        end generate AND3;

        AND4 : for i in 0 to 3 generate
        AND_4 : andGate port map (  A => a_reg(i), B => b_reg(3),F => AND_out (i +
            12));
        end generate AND4;

------FIRST OUTPUT ASSIGNMENT----%%%%
        z_mul_c(0) <= AND_out (0);

stage_4: fulladd port map (Cin => W(0), a => AND_out (1), b =>
    AND_out (4), sum => z_mul_c(1), Cout => AND_out (17));
stage_5: fulladd port map (Cin => AND_out (17), a => AND_out (2), b => AND_out (5)
    , sum => AND_out (30), Cout => AND_out (18));
stage_6: fulladd port map ( Cin => AND_out (18), a => AND_out (3), b => AND_out (6)
    , sum => AND_out (31), Cout => AND_out (19));
stage_7: fulladd port map ( Cin => AND_out (19), a => W(1), b => AND_out (7)
    , sum => AND_out (32), Cout => AND_out (20));

stage_8: fulladd port map (Cin => W(2), a => AND_out (30),b =>
    AND_out (8), sum => z_mul_c(2), Cout => AND_out (22));
stage_9: fulladd port map (Cin => AND_out (22), a => AND_out (31),b => AND_out (9),
     sum => AND_out (34),Cout => AND_out (23));
stage_10: fulladd port map (Cin => AND_out (23), a => AND_out (32),b => AND_out (10)
    ,sum => AND_out (33),Cout => AND_out (24));
stage_11: fulladd port map (Cin => AND_out (24), a => AND_out (20),b => AND_out (11)
    ,sum => AND_out (35),Cout => AND_out (25));

stage_12: fulladd port map (Cin => W(3), a => AND_out (34),b =>
    AND_out (12),sum => z_mul_c(3),Cout => AND_out (27));
stage_13: fulladd port map (Cin => AND_out (27), a => AND_out (33),b => AND_out
    (13),sum => z_mul_c(4),Cout => AND_out (28));
stage_14: fulladd port map (Cin => AND_out (28), a => AND_out (35),b => AND_out
    (14),sum => z_mul_c(5),Cout => AND_out (29));
stage_15: fulladd port map (Cin => AND_out (29), a => AND_out (25),b => AND_out
    (15),sum => z_mul_c(6),Cout => z_mul_c(7));


end Architecture simple_design_hier_structure;
```

A.3. TIMER UNIT

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
```

```
entity timer_unit is
  port (clk_timer                              : in std_logic; ----------------50
      MHz clock is considered for counter calculation
                reset_timer                          : in std_logic;
                timer_enable                         : in std_logic;
                select_timer                         : in std_logic_vector(2
                    downto 0);
        Monitor_signal                       : out std_logic
    );
end timer_unit;

architecture Behavioral of timer_unit is
type state_type is (st1_stop,st2_counter,st3_loop);----Finite state machine states

signal state: state_type:= st2_counter;

signal count                    : std_logic_vector(31 downto 0);
signal select_timer_flag        : std_logic;
signal select_timer_en          : std_logic;

begin
timer_block: process(clk_timer,reset_timer)
begin

if(reset_timer ='0') then
                count <= (others=>'0');

                select_timer_flag <= '0';
                select_timer_en   <= '0';

        state <= st2_counter;

        elsif (clk_timer'event and clk_timer ='1') then

                if (timer_enable = '1') then

                        case state is

                        when st1_stop =>

                                if (select_timer_en = '1' and select_timer = x"0")
                                    then
                                        count <= (others=>'0');
                                        select_timer_flag <= '0';
                                        --select_timer_en  <= '1';
                                        state <= st1_stop;

                                elsif ( select_timer_en = '1' or select_timer = x"1"
                                     or select_timer = x"2" or select_timer = x"3" or
                                     select_timer = x"4"
                                                or select_timer = x"5" or
                                                    select_timer = x"6" or
                                                    select_timer = x"7") then

                                        select_timer_en  <= '0';
                                        state <= st2_counter;
```

```
                end if;


            when st2_counter =>
            if( select_timer_en  = '0') then

                if (count >= x"0" and count < x"3D08E" and
                    select_timer = x"7" ) then ------5us timer  50MHz
                    clock is considered for counter calculation
                        count <= count+1;
                        state <= st2_counter;

                elsif (count >= x"0" and count < x"30D3F" and
                    select_timer = x"6" ) then ------4us timer 50MHz
                    clock is considered for counter calculation
                        count <= count+1;
                        state <= st2_counter;

                elsif ( count >= x"0" and count < x"249EF" and
                    select_timer = x"5" ) then ------3us timer 50MHz
                    clock is considered for counter calculation
                        count <= count+1;
                        state <= st2_counter;

                elsif (count >= x"0" and count < x"1869F" and
                    select_timer = x"4" ) then ------2us timer 50MHz
                    clock is considered for counter calculation
                        count <= count+1;
                        state <= st2_counter;

                elsif ( count >= x"0" and count < x"C34F" and
                    select_timer = x"3") then ------1us timer 50MHz
                    clock is considered for counter calculation
                        count <= count+1;
                        state <= st2_counter;

                elsif (count >= x"0" and count < x"61A7" and
                    select_timer = x"2" ) then ------500ns timer 50MHz
                     clock is considered for counter calculation
                        count <= count+1;
                        state <= st2_counter;

                elsif ( count >= x"0" and count < x"30D3" and
                    select_timer = x"1" ) then ------250ns timer 50MHz
                     clock is considered for counter calculation
                        count <= count+1;
                        state <= st2_counter;

                elsif ( select_timer = x"0" ) then
                        count <= (others=>'0');
                        select_timer_en  <= '1';
                        state <= st1_stop;

                else

                        select_timer_flag <= '1';
```

```
                                        --select_timer_en  <= '1';
                                        state <= st3_loop;
                                end if;
                        end if;

                        when st3_loop =>

                                if( select_timer_flag = '1') then
                                        count <= (others=>'0');
                                        select_timer_flag <= '0';
                                        select_timer_en  <= '0';
                                        state <= st2_counter;
                                end if;

                        end case;

                end if;

end if; -- end of reset else

end process;

Monitor_signal <= select_timer_flag;

end Behavioral;
```

A.4. LFSR

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_std.all ;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity random is
    generic ( width : integer :=  4 );
port (
       clk : in std_logic;
       random_num1 : out std_logic_vector (width-1 downto 0);   --output vector
            random_num2 : out std_logic_vector (width-1 downto 0)   --output vector
    );
end random;

architecture Behavioral of random is
signal rand_temp : std_logic_vector(width-1 downto 0):=(width-1 => '1',others =>
    '0');
signal temp : std_logic := '0';

signal rand_temp1 : std_logic_vector(width-1 downto 0):=(width-1 => '1',others =>
    '0');
signal temp1 : std_logic := '0';

begin
process(clk)
-- signal rand_temp : std_logic_vector(width-1 downto 0):=(width-1 => '1',others =>
    '0');
-- signal temp : std_logic := '0';
```

```
begin
if(rising_edge(clk)) then
temp <= rand_temp(width-1) xor rand_temp(width-2);
rand_temp(width-1 downto 1) <= rand_temp(width-2 downto 0);
rand_temp(0) <= temp;
end if;
random_num1 <= rand_temp;
end process;



process(clk)
-- signal rand_temp1 : std_logic_vector(width-1 downto 0):=(width-1 => '1',others =>
     '0');
-- signal temp1 : std_logic := '0';
begin
if(rising_edge(clk)) then
temp1 <= rand_temp1(width-1) xor rand_temp1(width-3);
rand_temp1(width-1 downto 1) <= rand_temp1(width-2 downto 0);
rand_temp1(0) <= temp1;
end if;
random_num2 <= rand_temp1;
end process;

end architecture;
```

BIBLIOGRAPHY

[1] N. H. E. Weste and D. M. Harris, *Journal of Chemical Information and Modeling*, Vol. 53 (2013) arXiv:arXiv:1011.1669v3 .

[2] J. W. Tschanz, J. T. Kao, S. G. Narendra, R. Nair, D. a. Antoniadis, A. P. Chandrakasan, and V. De, *Adaptive body bias for reducing impacts of die-to-die and within-die parameter variations on microprocessor frequency and leakage,* IEEE Journal of Solid-State Circuits **37**, 1396 (2002).

[3] S. Das, S. Pant, D. Roberts, S. Lee, D. Blaauw, T. Austin, T. Mudge, and K. Flautner, *A self-tuning DVS processor using delay-error detection and correction,* IEEE Symposium on VLSI Circuits, Digest of Technical Papers **2005**, 258 (2005).

[4] D. Kit, *Analog Flow 28 UTBB-FDSOI Process Design Kit,* , 104 (2014).

[5] F. Worm, P. Ienne, P. Thiran, and G. De Micheli, *A robust self-calibrating transmission scheme for on-chip networks,* IEEE Transactions on Very Large Scale Integration (VLSI) Systems **13**, 126 (2005).

[6] E. Koutroulis, A. Dollas, and K. Kalaitzakis, *High-frequency pulse width modulation implementation using FPGA and CPLD ICs,* Journal of Systems Architecture **52**, 332 (2006).

[7] O. Trescases, G. Wei, and W. T. Ng, *A segmented digital pulse width modulator with self-calibration for low-power SMPS,* 2005 IEEE Conference on Electron Devices and Solid-State Circuits, EDSSC , 367 (2006).

[8] G. E. Moore, *Cramming more components onto integrated circuits,* Proceedings of the IEEE **86**, 82 (1998).

[9] R. H. Dennard, F. H. Gaensslen, H. N. Yu, V. L. Rideout, E. Bassous, and A. R. Leblanc, *Design of Ion-Implanted MOSFET's With Very Small Physical Dimensions,* IEEE Journal of Solid-State Circuits **9**, 256 (1974).

[10] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, a. Keshavarzi, and V. De, *Parameter variations and impact on circuits and microarchitecture,* Proceedings 2003. Design Automation Conference (IEEE Cat. No.03CH37451) **64**, 338 (2003).

[11] J. Srinivasan, S. S. V. Adve, P. Bose, and J. Rivers, *The case for lifetime reliability-aware microprocessors,* Proceedings. 31st Annual International Symposium on Computer Architecture, 2004. (2004), 10.1109/ISCA.2004.1310781.

[12] J. Srinivasan, S. V. Adve, P. Bose, and J. a. Rivers, *Lifetime reliability: Toward an architectural solution,* IEEE Micro **25**, 70 (2005).

[13] S. Das, C. Tokunaga, S. Pant, W.-h. Ma, S. Kalaiselvan, K. Lai, D. M. Bull, and D. T. Blaauw, *RazorII: In Situ Error Detection and Correction for PVT and SER Tolerance,* IEEE Journal of Solid-State Circuits **44**, 32 (2009).

[14] K. a. Bowman, J. W. Tschanz, N. S. Kim, J. C. Lee, C. B. Wilkerson, and S.-l. L. Lu, *Dynamic-Variation Tolerance,* (2008).

[15] N. Kamae, A. Tsuchiya, and H. Onodera, *An area effective forward/reverse body bias generator for within-die variability compensation,* IEEE Asian Solid-State Circuits Conference 2011 **2**, 217 (2011).

[16] M. Fojtik, D. Fick, Y. Kim, N. Pinckney, D. M. Harris, D. Blaauw, and D. Sylvester, *Bubble razor: Eliminating timing margins in an ARM cortex-M3 Processor in 45 nm CMOS using architecturally independent error detection and correction,* IEEE Journal of Solid-State Circuits **48**, 66 (2013).

[17] S. Nassif, K. Bernstein, D. J. Frank, A. Gattiker, W. Haensch, B. L. Ji, E. Nowak, D. Pearson, and N. J. Rohrer, *High Performance CMOS Variability in the 65nm Regime and Beyond mduernabria lwfaci ties The / dis,* Technology , 11 (2007).

[18] E. Malavasi, S. Zanella, J. Uschersohn, and M. Misheloff, *Impact analysis of process variability on digital circuits withperformance limited yield,* Statistical Methodology , 60 (2001).

[19] M. Zhang, T. M. Mak, J. Tschanz, K. S. Kim, N. Seifert, and D. Lu, *Design for resilience to soft errors and variations,* Proceedings - IOLTS 2007 13th IEEE International On-Line Testing Symposium , 23 (2007).

[20] B. C. Paul and T. Corp, *Circuit Failure Prediction and Its Application to Transistor Aging Mridul Agarwal Stanford University Intel Corporation Subhasish Mitra,* 25th IEEE VLSI Test Symposium (VTS'07) , 277 (2007).

[21] D. Ernst, N. S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, T. Mudge, B. Ave, and A. Arbor, *Razor : A Low-Power Pipeline Based on Circuit-Level Timing Speculation,* (2003).

[22] T. Sato and Y. Kunitake, *A simple flip-flop circuit for typical-case designs for DFM,* Proceedings - Eighth International Symposium on Quality Electronic Design, ISQED 2007 , 539 (2007).

[23] T. Sakurai, A. Matsuzawa, and T. Douseki, *Fully-depleted SOI CMOS circuits and technology for ultra-low power applications* (Springer 2006).

[24] X. Cauchy and F. Andrieu, *Questions and answers on Fully Depleted SOI technology,* SOI consortium white paper , 1 (2010).

[25] Taur and T. H. Ning., *Journal of Chemical Information and Modeling*, Vol. 53 (1998) arXiv:arXiv:1011.1669v3 .

[26] J. Baker, *CMOS Circuit Design, Layout, and Simulation*, Vol. XXXIII (2012) pp. 81–87, arXiv:9809069v1 [arXiv:gr-qc] .

[27] S. L. Lu, *Speeding up processing with approximation circuits,* Computer **37**, 67 (2004).

[28] S. L. Lu, *Speeding up processing with approximation circuits,* Computer **37**, 67 (2004).

[29] S. Das, *Razor: A Variability-Tolerant Design Methodology for Low-Power and Robust Computing.* (2009).

[30] R. Hegde and N. R. Shanbhag, *A voltage overscaled low-power digital filter IC,* IEEE Journal of Solid-State Circuits **39**, 388 (2004).

[31] H. Fuketa, M. Hashimoto, Y. Mitsuyama, and T. Onoye, *Adaptive performance compensation with in-situ timing error predictive sensors for subthreshold circuits,* IEEE Transactions on Very Large Scale Integration (VLSI) Systems **20**, 333 (2012).

[32] S. Hanson, M. Seok, Y. S. Lin, Z. Foo, D. Kim, Y. Lee, N. Liu, D. Sylvester, and D. Blaauw Dr., *A low-voltage processor for sensing applications with picowatt standby mode,* IEEE Journal of Solid-State Circuits **44**, 1145 (2009).

[33] M. Meijer, J. P. de Gyvez, B. Kup, B. van Uden, P. Bastiaansen, M. Lammers, and M. Vertregt, *A forward body bias generator for digital CMOS circuits with supply voltage scaling,* Proceedings of 2010 IEEE International Symposium on Circuits and Systems , 2482 (2010).

[34] A. Srivastava and C. Zhang, *An Adaptive Body-Bias Generator for Low Voltage CMOS VLSI Circuits,* International Journal of Distributed Sensor Networks **4**, 213 (2008).

[35] C. H. Kim and K. Roy, *Dynamic VTH scaling scheme for active leakage power reduction,* Proceedings -Design, Automation and Test in Europe, DATE , 163 (2002).

[36] T. Kuroda, K. Suzuki, S. Mita, T. Fujita, F. Yamane, F. Sano, A. Chiba, Y. Watanabe, K. Matsuda, T. Maeda, T. Sakurai, and T. Furuyama, *Variable supply-voltage scheme for low-power high-speed CMOS digital design,* IEEE Journal of Solid-State Circuits **33**, 454 (1998).

[37] T. Seki, S. Akui, K. Seno, M. Nakai, T. Meguro, T. Kondo, A. Hashiguchi, H. Kawahara, K. Kumano, and M. Shimura, *Dynamic voltage and frequency management for a low-power embedded microprocessor,* IEICE Transactions on Electronics **E88-C**, 520 (2005).

[38] H. Fuketa, M. Hashimoto, Y. Mitsuyama, and T. Onoye, *Adaptive performance compensation with in-situ timing error predictive sensors for subthreshold circuits,* IEEE Transactions on Very Large Scale Integration (VLSI) Systems **20**, 333 (2012).

[39] T. Nakura, K. Nose, and M. Mizuno, *Fine-grain redundant logic using defect-prediction flip-flops,* Digest of Technical Papers - IEEE International Solid-State Circuits Conference , 21 (2007).

[40] H. Fuketa, M. Hashimoto, Y. Mitsuyama, and T. Onoye, *Trade-off analysis between timing error rate and power dissipation for adaptive speed control with timing error prediction,* 2009 Asia and South Pacific Design Automation Conference , 3094 (2009).

[41] J. Mauricio and F. Moll, *Local variations compensation with DLL-based Body Bias Generator for UTBB FD-SOI technology,* Conference Proceedings - 13th IEEE International NEW Circuits and Systems Conference, NEWCAS 2015 (2015), 10.1109/NEWCAS.2015.7182005.

[42] D. Pappalardo and I. Introduction, *Charge Pump Circuits : An Overview on Design Strategies and Topologies,* Circuits and Systems Magazine, IEEE, **10**, 31 (2010).